A DEEP LEARNING-BASED HYBRID COMPUTATIONAL APPROACH TO CARDIAC ELECTROPHYSIOLOGY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALI FATIH KULOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

SEPTEMBER 2023

Approval of the thesis:

A DEEP LEARNING-BASED HYBRID COMPUTATIONAL APPROACH TO CARDIAC ELECTROPHYSIOLOGY

submitted by **ALI FATIH KULOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Erdem Canbay
Head of Department, **Civil Engineering** _____

Assoc. Prof. Dr. Serdar Göktepe
Supervisor, **Civil Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Ercan Gürses
Department of Aerospace Engineering, METU _____

Assoc. Prof. Dr. Serdar Göktepe
Department of Civil Engineering, METU _____

Assoc. Prof. Dr. Hamdullah Yücel
Institute of Applied Mathematics, METU _____

Assist. Prof. Dr. Şeyma Bilazeroğlu
Department of Mathematics, Çankaya University _____

Assist. Prof. Dr. Ferit Sait
Department of Aerospace Engineering, Atılım University _____

Date:11.09.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Ali Fatih Kuloğlu

Signature :

ABSTRACT

A DEEP LEARNING-BASED HYBRID COMPUTATIONAL APPROACH TO CARDIAC ELECTROPHYSIOLOGY

Kuloğlu, Ali Fatih
M.S., Department of Civil Engineering
Supervisor: Assoc. Prof. Dr. Serdar Göktepe

September 2023, 106 pages

Electrophysiological modeling of the heart has witnessed significant progress with the increase of available computational power. Realistic electrophysiology models often require the solution of highly nonlinear differential equation systems. The complex nature of the electrodynamic activity of a cell limits the applicability of simplistic numerical techniques and necessitates the utilization of more advanced and demanding techniques. Deep learning has emerged as a promising tool for predicting the solution of highly nonlinear problems and has shown tremendous success in differential equation-based phenomena of biological systems over recent years. In this work, a deep learning-based algorithm is proposed for the accurate and time-efficient solution of the electrophysiology problem of the heart. A deep learning-based model is developed for forecasting transmembrane voltage at the cellular level. For this purpose, the biophysically detailed ten Tusscher-Panfilov model is used for the generation of the training data and performance measurements. Training data are acquired by solving ten Tusscher-Panfilov model as an ordinary differential equation system. The resulting deep learning-based model incorporates the external stimulus information and past potential values while making predictions. An important novelty of this

work is extending a model trained with ordinary differential equations to the realm of partial differential equations by associating the external stimuli with the conductivity term of the partial differential equation. This approach facilitates the application of more conventional partial differential equation solvers. Therefore, the classical way of solving partial differential equations is combined with deep learning in the proposed approach. This hybrid approach has successfully been applied to solve multiple problems and has been evaluated in different settings.

Keywords: Deep Learning, Cardiac Electrophysiology, Finite Element Modeling

ÖZ

KALP ELEKTROFİZYOLOJİSİNE DERİN ÖĞRENME TABANLI HİBRİD BİR HESAPLAMALI YAKLAŞIM

Kuloğlu, Ali Fatih
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Serdar Göktepe

Eylül 2023 , 106 sayfa

Kalp elektrofizyolojisinin modellenmesinde, mecvut hesaplama gücündeki artışla önemli ilerlemeler sağlanmıştır. Gerçekçi elektrofizyoloji modelleri genellikle yüksek derecede karmaşık olan diferansiyel denklem sistemlerini çözmeyi gerektirmektedir. Kalp hücresinin elektrodinamik aktivitesinin karmaşık doğası, basit sayısal tekniklerin uygulanabilirliğini sınırlamakta ve daha karmaşık sayısal tekniklerin kullanımını zorunlu hale getirmektedir. Derin öğrenme, son yıllarda biyolojik sistemlerin diferansiyel denklem temelli problemlerinde başarıyla kullanılmış, doğrusal olmayan problemleri tahmin etmek için umut vaat eden bir araç olarak ortaya çıkmıştır. Bu çalışmada, kalbin elektrofizyoloji probleminin doğru ve hızlı bir şekilde çözülmesi için derin öğrenme tabanlı bir algoritma önerilmektedir. Derin öğrenme modeli, transmembran voltajını hücre ölçeğinde tahmin etmek için geliştirilmiştir. Modelin öğrenimi için, biyofiziksel ayrıntıları içeren ten Tusscher-Panfilov modeli kullanılmış ve performans ölçümleri bu modelin sonuçları üzerinden yapılmıştır. Öğretme verileri, ten Tusscher-Panfilov modelinin adi diferansiyel denklem sistemi olarak çözülmesiyle elde edilmiştir. Elde edilen model, dışarıdan gelen uyarıları ve geçmiş potansi-

yel değerleri, beklenen potansiyelin değerini tahmin etmek amacıyla kullanmaktadır. Bu çalışmanın önemli bir yeniliği, adi diferansiyel denklemler kullanılarak geliştirilmiş olan bir modelin, dışarıdan gelen uyarıları iletkenlik terimiyle ilişkilendirerek kısmi diferansiyel denklemleri çözebilecek bir şekilde genişletilmesidir. Bu yaklaşım, daha konvansiyonel kısmi diferansiyel denklem çözme yollarının kullanımına da olanak sağlamaktadır. Böylece, kısmi diferansiyel denklemlerinin çözülmesinde kullanılan klasik metotlar ile derin öğrenme yaklaşımının beraber çalışması sağlanmıştır. Bu hibrit yaklaşım, birden fazla problemi başarıyla çözmüş ve farklı şartlar altında değerlendirilmiştir.

Anahtar Kelimeler: Derin Öğrenme, Kalp Elektrofizyolojisi, Sonlu Eleman Modellemesi

Dedicated to my dear parents.

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my thesis supervisor, Assoc. Prof. Serdar Göktepe. His guidance, perspective, knowledge, and insight made this study possible. He set an example for me not only as a scientist but also with his humanity. His support and guidance helped me to decide on my future goals. I feel lucky to have had the opportunity to work with him during my studies.

Furthermore, I would like to thank the members of my thesis examining committee, Assoc. Prof. Ercan Gürses, Assoc. Prof. Hamdullah Yücel, Assist. Prof. Şeyma Bilazeroğlu and Assist. Prof. Ferit Sait for their contributions and insights.

I would like to express my gratitude to Fikret Küçükdeveci and my colleagues at Kardinero for their support and for providing an excellent working environment.

I would like to thank Sinan Fırat Dal and Özgür Pasaoglu for their assistance during my graduate studies.

I would like to thank my dearest friends, Mertcan Özel, Serhat Uzbaş, and Hazal Özen, for their support, understanding, and friendship throughout all these years.

Lastly, I would like to thank my father, Süleyman Sırrı Kuloğlu, and my mother, Mukadder Kuloğlu, for their unconditional love, support and encouragement throughout my life. I cannot express how grateful I am to have such parents. No words can express my love for them. Also, I would like to thank my girlfriend and best friend, Sevda Rafatova, for always being there for me. Her support, patience, and love helped me overcome many challenges. I am very thankful that I met her in my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
CVD	Cardiovascular Disease
OECD	The Organization for Economic Cooperation and Development
MRI	Magnetic Resonance Imaging
MR	Magnetic Resonance
ML	Machine Learning
DL	Deep Learning
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
DNN	Deep Neural Network
PINN	Physics-Informed Neural Networks
P-DL	Physics-Constrained Deep Learning

CHAPTER 1

INTRODUCTION

In this thesis, the aim is to develop a new and practical approach to the numerical solution of the electrophysiology problem of the human heart. The proposed approach in this thesis is based on deep learning. A well-accepted high fidelity electrophysiological model is used to generate the necessary data for training the deep learning model. The electrophysiological problem is solved using this trained deep learning model in two dimensional space with Finite Element Analysis Program (FEAP). The results of these solutions are evaluated and discussed. The performance of the proposed deep learning model is compared with the high-fidelity model's performance.

This chapter presents a structured assessment of the focused problem, the aims of the thesis, and why this study is needed. First, the motivation of the thesis is given. After the motivation, the addressed problem in this study is explained in more detail. In the literature review section, the current state of the literature is presented, and recent advances in solving similar problems are reviewed. Related studies are briefly summarized to show the novelty of the proposed approach. Next, the aim and the intended outcomes of the thesis are explained. The last part states the different aspects of the problem that will be covered in this study.

1.1 Motivation

The human heart has been a major interest in science for centuries. Researchers have studied the underlying mechanisms of the heart for a long time. As a result of these studies, the chemical, mechanical, and biological processes of the heart have been largely understood. Mathematical tools have been an essential part of this progress.

These tools help researchers immensely to model and understand complex underlying mechanisms of the heart. The significant surge of computational power in equipment in recent decades has increased the utilization of mathematical tools, and these new tools have become major aspects of this field. More complex and applicable mathematical models led to the creation of more realistic and consistent studies and solutions to more complex problems. In the end, the dissemination of these new studies helps the creation of new medical procedures and improves understanding of the heart. The development of new and better mathematical models is still a critical necessity since cardiovascular diseases (CVDs) are one the most prominent global problems today.

CVDs cause the most significant number of deaths compared to other medical conditions globally. In 2019, approximately 17.4 million people died from CVDs. This number represents approximately 32% of global deaths. Noncommunicable diseases caused 17 million premature deaths (under the age of 70) in 2019, and 38% of these deaths occurred due to CVDs [4]. CVDs cause the highest number of deaths among noncommunicable diseases, followed by cancer, upper respiratory diseases, and diabetes [5]. According to the European Heart Network statistics from 2017, CVDs are causing 3.9 million deaths in Europe and over 1.9 million deaths in the European Union (EU). In Europe, CVDs cause 45% of deaths, while in the EU, this rate is around 35% [6]. In England, CVDs cause approximately 136,000 deaths per year, which is around 25% of the total number of deaths. CVDs are the primary reason for the total morbidity, disability, and mortality in England [7]. Finally, in the United States, CVDs caused 695,000 deaths in the year 2021, and they are the primary cause of death. This number is around 20% of total deaths in the United States [8].

CVDs also impose a significant financial burden on the economy. CVDs are estimated to cost €210 billion to the EU economy yearly in 2017 [6]. The economic impact on the US economy during the period from 2018 to 2019 amounted to $407.3 billion, consisting $251.4 billion in direct costs and $ 155.9 billion in indirect costs, including factors such as productivity loss and mortality [9]. It is expected for these costs to rise and, by the year 2035, to reach $1.1 trillion per year [10]. In England, CVDs cost £7.8 billion to the healthcare system and £15.8 billion to the economy. Furthermore, 1 in 6 people is predicted to have a stroke in their lifetime, and social services for stroke

survivors cost around £5.2 billion per year [7]. In 11 OECD member countries, CVDs account for 17% of hospital expenditure, with the largest share of hospital spending [11]. CVDs having substantial adverse impacts on economies is an indisputable fact, according to the findings of multiple studies; see, e.g., [6, 7, 9, 10, 11].

In the United States, the prevalence of CVDs for people aged 20 and above is around 48%. If hypertension is not included, this rate is around 9.9% [9]. Further, it is predicted that by 2035, around 45% of the US population will have some form of CVD [10]. Although the European Society of Cardiology member countries have observed a slight decrease in the incidence of CVD cases, in some middle-income countries, CVD cases increased modestly [12].

It is logical to expect CVDs to continue as a major medical concern. The decrease in the mortality rate has stalled in recent years, and CVD cases are expected to surge in the near future, according to multiple statistical studies [10, 12]. In this stage, new mathematical approaches and the utilization of recent technologies are needed to model more complex mechanisms of the heart in a faster and more effective way. These new modeling methods will lead to medical advances ensuring effective combat against CVDs.

1.2 Problem Statement

The human heart is a vital organ encompassing specific tissues and unique mechanisms. Therefore, it offers different problems than the rest of the human body. Cardiac electrophysiology has been a focus of many studies in the past; see, e.g., [13, 14, 15]. The electrophysiology of the heart is directly coupled with its mechanical behavior. Correct modeling of electrophysiology can lead to a better understanding of medical conditions and the development of new medical procedures.

Electrophysiology of the heart is a complex phenomenon. Multiple factors play an essential role in electrophysiological behavior. Ion concentrations inside and outside the cells, geometric characteristics of the heart, orientation and distribution of tissue fibers, and distribution of various cell types constitute important factors in electrophysiology. The scale of these factors differs substantially. Calculating changes in

the ion concentrations and cell gate states are problems that need to be solved on the microscale. On the other hand, the propagation of electrical waves in the heart, with the heart's unique geometry, fiber orientation, and distribution, is solved on the macroscale.

Solving this electrophysiological problem in a correct way considering both macroscale and microscale aspects is possible, and multiple different approaches exist in the literature. Phenomenological modeling and ionic modeling are two main approaches for obtaining a solution. The phenomenological models are based on observations of electrophysiological behavior on the microscale [16], while the ionic models take microscale variables into consideration directly [13]. In general, phenomenological models provide faster solutions. However, these solutions do not give physically meaningful quantitative results on the microscale. Physical quantities on the microscale are often crucial for scientific observations and analysis for further studies. Phenomenological results are also less accurate than results obtained from ionic models in general. Ionic models provide physically accurate results at both scales. These models generally involve more variables and a higher number of equations that need to be solved. As a result, the solution process may require a substantial amount of time and computational power. Time and computation power restraints prevent researchers with limited resources from employing more sophisticated and accurate results and create a bottleneck for scientific progress.

1.3 Literature Review

There has been an immense increase in the computational power and capabilities of computational tools over the past years. This increase has resulted in the solution of numerous challenging problems and rapid progress in almost every field of science. Thus, newer and more complex problems have started to be focused on due to this progress. However, in recent years, advances have slowed down as solution approaches encountered bottlenecks. These complex problems require more advanced algorithms or exceed time and resource constraints. This situation eventually led to the utilization of deep learning. Although deep learning has been around for a long time, many applications of it are relatively new.

Cardiology and computational science are two of the major fields that made use of deep learning. With the introduction of deep learning in the last few years, a significant amount of literature has been published in both fields; see, e.g., [17, 18, 19, 20]. Deep learning offers a potential for further advancements, as this completely new approach has shown that it can quickly solve highly complex problems. A wide range of deep learning studies focus on cardiology in the literature. Hospital cardiac-arrest prediction [18] and classification of heart sound signals [21] are examples of DL-based studies in cardiology. Moreover, DL is also utilized with common diagnostic tools for cardiac disease detection, such as magnetic resonance imaging (MRI) [22, 23, 24], and electrocardiogram (ECG) [25, 26, 27, 28, 29].

The electrophysiological behavior of the heart is a mathematically complex problem. Understanding the electrophysiological behavior and successful computational modeling of underlying mechanisms requires solutions of large differential equation systems in general. As a consequence, solving these complex differential equation systems can be computationally extremely expensive and time-consuming. DL is an emerging, promising tool for solving ordinary and partial differential equations; see, e.g., [20, 30, 31]. In 1994, A. J. Meade proposed a method for solving ODEs using neural networks [19]. This study proposed a feed-forward neural network and constraints on network weights and basis functions. A family of neural network models was introduced for solving ODEs by Chen et al. in [20]. Instead of specifying a discrete sequence of hidden layers, the proposed approach employed a neural network to parameterize the derivative of the hidden state. In this model, gradients are computed using the adjoint sensitivity method, and the network output is computed using a black-box differential equation solver. However, most biological system equations are represented by stiff equations; as a result, explicit solutions are unstable, and the solution process is computationally costly. Thus, neural ODEs tend to fail in these kinds of problems. These problems can be overcome by employing non-reserving adjoint methods, linear scaling of adjoints, and the scaling of the equations [32]. Another numerical framework for approximating governing equations of ODEs is presented in [33]. In this work, residual network-based deep neural networks (DNN) approximate governing equations and integrate them in time with the given initial condition. Results showed that DNNs are highly capable of governing equation ap-

proximation. This work extended in [34], aiming to approximate complex biological processes and uncover their unknown factors. Additional variables are incorporated into the DNN model; some variables are temperature and electric current, which can be controlled during experimentation. Their incorporation enables the investigation of the influence of the controlled variable in the dynamics of the system and results in a more flexible network to understand underlying dynamics.

Analytical solutions for PDEs can be difficult to obtain, and most of the time, obtaining an analytical solution might not be possible. Numerical approaches frequently face challenges when dealing with PDEs, such as the possibility for numerical instability, the potential for extremely high dimensionality, and the difficulty of capturing complex phenomena. One DL-based approach for solving ODEs and PDEs is proposing a trial solution, such that the trial solution satisfies initial value and boundary conditions by definition. A DNN approximation function is placed in the trial solution and trained to learn from data [35]. In recent years, the data-driven discovery of PDEs was proposed based on sparse regression technique [36]. However, this technique assumes the chosen library to be sufficiently rich in reflecting the time dynamics of the problem, which is unlikely and requires more data points than DL-based techniques. The Deep Ritz Method is a modern approach for solving variational problems that arise from PDEs with DL [37]. This method is based on representing the functions in the context of the Ritz method and minimizing the variational problem with known data. It is an adaptive method and less sensitive to dimensionality. The Deep Galerkin Method is another approach for solving high-dimensional PDEs. Importantly, it is a meshless method aiming to reduce the computational cost of dimensionality. The solution is approximated with a deep neural network, which is trained to satisfy the differential operator, initial condition, and boundary conditions. The algorithm generates random points with probability density function and is trained by using these points [38]. Encoding underlying physical laws of PDEs into neural networks (NNs) is an idea utilized by physics-informed neural networks (PINNs) [31]. Known physical laws are explicitly stated in the training of the NN, and a solution approximator model is created. PINNs can be used in continuous and discrete forward solutions and are capable of data-driven discovery of PDEs. Differentiation can be performed with these nonlinear solutions after approximation of the solution function with PINNs.

Automatic differentiation is the key concept for finding derivatives, and it is superior and computationally more efficient than numerical differentiation most of the time. Multiple well-known differential equations such as Navier-Stokes, Nonlinear Schrödinger, and Burger's equation were solved using PINNs and produced satisfactory results [30]. It is shown that increasing the number of training points in the high residual region in the domain when training a PINN can improve the model's efficiency [39]. While the global minimization approach works well for boundary value conditions of the problem, catastrophic forgetting, a phenomenon observed in neural networks where they tend to forget previously acquired sequential information, can make these PDE-solving techniques inapplicable to time-dependent phenomena. A method is introduced for solving initial value PDEs, aiming to mitigate the initial value forgetting problem, and multiple techniques are investigated, including; improvements on representational power, scalability, stability, and conditioning [40]. In [41], a method for solving the parabolic PDEs is investigated. Nonlinear parabolic PDEs are reformulated as backward stochastic differential equations, and the gradient of the solution is approximated using DNNs. This approach is shown to solve high dimensional PDEs such as the Schödinger equation and the nonlinear Black-Scholes equation.

Reaction-diffusion problems describes the dynamics of the system by considering the spread and interactions of a quantity in space and time. Diffusion refers to the spread of matter from high-concentration regions to low-concentration zones, while reaction refers to the transformation of matter. The cardiac electrophysiology problem is a reaction-diffusion type of problem. In [42], an encoder-decoder-based convolutional NN is designed to predict the concentration of a matter. When boundary conditions, time, diffusion coefficient, and the reaction rate are given as input to this model, it produces acceptable results in a faster manner than the traditional finite element method. However, the model becomes unstable with the increase in inputs and complex geometrical configurations. Physics-encoded Recurrent Convolutional Neural Network (PeRCNN) aims to hard code physical constraints of the systems directly into the model, unlike PINNs [43]. PeRCNN is a discrete mesh-based model with a unique convolutional layer to increase effectiveness for nonlinear spatiotemporal dynamics, and it can incorporate numerical time integration methods. PeRCNN outputs good re-

sults for multiple reaction-diffusion problems, but the model is designed for standard convolution operators, limiting its applicability to irregular geometries. Moreover, underlying governing PDEs are assumed to have only polynomial forms. In [44], a DL model with U-Net-based architecture solves a one-dimensional reaction-diffusion problem without any labeled information but only using initial and boundary conditions. This proposed approach will likely encounter problems in complex geometries such as the heart.

Complex systems involve interactions and processes in different scales or levels of detail. In many mathematical systems, underlying processes of different scales are linked to each other, and multiscale modeling aims to bridge the gap between these scales. Thus, multiscale modeling is needed in numerous biological, engineering, and chemical systems to understand underlying phenomena and represent system behavior. Overall, DL has a limited application in many aspects of multiscale modeling; however, DL techniques carry a high potential to address the known limitations of the current state of modeling [45]. Homogenization of information from the microscale to the macroscale is an important opportunity for DL in multiscale modeling. An exemplary work for utilization of DL is a multiscale simulation of flow dynamics [46]. NN combines observed fine data and physical concepts with local multiscale model reduction methodologies to predict flow dynamics.

Pure DL models learn only using the available data without prior knowledge about underlying physical laws and constraints. Solving the problem in accordance with the physical laws realistically carries significant importance in heart-related mathematical problems. If a model does not abide by the boundary conditions and governing equations to some degree and only learns from data, it is often inapplicable for scientific studies. It is possible to predict arterial blood pressure from non-invasive 4D flow MRI data by using PINNs [17]. Mentioned PINNs effortlessly integrate non-invasive in-vivo measurements and computational flow dynamics models resulting from physical equations. Conservation of momentum, conservation of mass, losses at the interfaces, and clinical measurement results are accounted for in the given model, and it is shown that the model is highly accurate. Since flow conditions do not change greatly with different patients, trained networks can be adapted to different cases. However, there is still a need to reduce the problem setup and training time for clini-

cal applications. A DL framework with physical constraints (P-DL) for inverse ECG modeling was proposed to take physics into account [47]. The proposed framework integrates the physical laws for cardiac electrical wave propagation with DL infrastructure to predict spatiotemporal electrodynamics using electric potentials measured by the body-surface sensor network. The resulting network's accuracy is remarkably higher than that of the model that does not know about physical constraints. An important point to note is that incorporating physics knowledge into the model is a simple process, and DL provides computational efficiency compared to traditional methods. Inference regarding ventricular activation parameters can be performed with DL [48]. This patient-specific model locates Purkinje endocardial root nodes and predicts conduction velocities using ECG and cardiac magnetic resonance imaging data and patient-specific inputs, such as age, sex, and body mass index. The encoder-decoder structure of the model framework enables the model to first encode ECG and cardiac magnetic resonance imaging data, and then by decoding the given data, the model is trained to generate geometrically and physically possible results. PINN-based cardiac activation mapping is another method for cardiac activation mapping [49]. In this NN, cardiac activation mapping that accounts for the wave propagation dynamics is predicted. The physical part of the neural network takes the Eikonal equation into account, and epistemic uncertainty is computed based on the predicted results. The model successfully predicted activation mappings in atria within a range of specified uncertainty.

Physical processes responsible for action potential in cardiac cells are generally described as a system of ODEs. However, computationally, modeling the whole heart at the cellular level is not possible. Homogenization of this discrete cell model leads to a continuum model in the form of PDEs. Therefore, the cardiac electrophysiology problem requires a solution of PDEs in heart tissue, and it is often characterized as a multiscale modeling problem. DNNs can be trained to make predictions for a dynamic system such as cardiac electrophysiology. Sequential frames from the dynamic system can be used to predict the next frame of a domain using recurrent DNNs [50]. By modifying the P-DL framework to integrate the physical laws of cardiac electrical wave propagation for robust prediction of heart electrical behavior from sparse sensor measurements, a physics-constrained deep active learning (P-DAL) framework is

proposed [51]. A drop-out is used for uncertainty estimation, and geodesic distance-based space-filling is used for distributing the data points over the domain. As a result, the model learns from less data compared to the randomly distributed data over the domain and manages to capture the electrodynamic behavior of the heart with high accuracy. EP-PINNs can accurately predict electrical wave propagation in the heart tissue, action potential, and electrophysiological parameter estimation [52]. After training with sparse in-silico or in-vitro data, the proposed NN accurately captures the spatiotemporal evolution of action potential, predicts excitability, diffusion, and action potential duration parameters, and identifies heterogeneities in the network parameters.

PINN-based fully connected NNs and recurrent NNs for solving cardiac electrophysiology model, namely, the Fitzhhugh-Nagumo model, are compared in [53]. PINN-based designs resulted in better predictions for both architectures, and fully connected NNs outperformed recurrent NNs. Data-driven approaches are capable of designing monodomain models with the usage of artificial NNs [54]. In this work, the classical way of determining coefficients of a predefined model is replaced by designing the model with artificial NN and a model for phenomena related to cardiac electrophysiology designed using optimal control techniques. Spatio-temporal dynamics of the cardiac tissue can be directly predicted [55]. Necessary data for training is generated with a model that describes electrical activation in cardiac tissue. Electrical dynamics can be directly predicted with this model with high accuracy. However, for the long-term prediction process, accuracy drops significantly. In [56], an approach to replace the numerical integration of PDEs with a DL-based algorithm named EP-NET is proposed. The dynamic system is reformulated, and integration is performed with NN-based operators. After the model assimilates the first few frames of the domain, the evolution of the model is predicted. Later, EP-NET extended to learn cardiac electrophysiology dynamics from the data when scars are present in the cardiac tissue [57]. The model successfully generalized to unseen conditions and learned complex initial and boundary conditions. Still, the EP-NET model faces some problems; this data-driven model could not successfully reproduce complex repolarization dynamics of cardiac tissue, and prediction performance gradually declined [58]. To alleviate the issues of EP-NET, a two-component architecture network is proposed. The net-

work splits the dynamics of the system into physics-driven and data-driven parts. The proposed network correctly reflected spatiotemporal electrodynamic behavior in the domain when trained with the ten Tusscher-Panfilov model. The capabilities of this framework explored in regards to learning full cardiac potential cycle [59]. Also, the framework has the ability to produce complex action potential dynamics such as depolarization and repolarization phases in the presence of noise in the data. In general, the solution of coupled nonlinear systems regarding electrophysiology relies on numerical approximation. Reducing these systems to lower dimensional problems by conventional reduced-order models is challenging. A DL-based framework for reduced order modeling of nonlinear time-dependent parametrized PDEs named DL-ROM is proposed to solve this issue [60]. In the framework, nonlinear trial manifold and nonlinear reduced dynamics are learned by the DL model in a non-intrusive way. DL-ROM framework can efficiently provide solutions to parametrized electrophysiology problems with a high accuracy [61]. One downside of training DL-ROMs is extremely long learning times. DL-ROM framework is extended to address the bottleneck in the training process [62]. DL-ROM framework is modified by performing prior dimensionality reduction on full-order models through proper orthogonal decomposition. This process enables speed up training times and decreases network complexity. The proposed framework, named POD-DL-ROM, can solve challenging physiological and pathological electrophysiology problems in real-time.

In the context of ODEs, taking advantage of known electrophysiological behavior and underlying variables of an electrophysiological model is possible. However, the complex nonlinear behavior of the electrical response in cardiac cells imposes a significant challenge. The prediction of the electrical voltage of a cardiac cell by an NN is an achievable task [63]. The reason is that NNs are extremely effective in pattern recognition tasks, and recurrent NNs have the ability to keep track of the previous patterns. As a result, recurrent NNs can learn the patterns of generated electrical signals in the heart. NN predictions can physiologically reflect the behavior of the model used in training, with the potential for decreased computation time and high accuracy. Different NN architectures have been proposed for nonlinear time series prediction of cardiac voltage data, and their performance has been compared in the literature [64]. An important result of adapting time series-based prediction models is that they can

predict potential values consistently for a long time with a high degree of accuracy. In this work, echo state networks provide the best accuracy with the lowest computational time; however, echo state networks are very sensitive to hyperparameters and network parameters. In the study [65], the echo state framework in [64] is extended with the integration of a long-short term memory autoencoder. The autoencoder is integrated to represent input nonlinearities in a compressed form, extract features for more robust predictions, and simplify the learning process. This data-driven approach is computationally more demanding than a pure echo state network. However, it is more robust and accurate.

In the current state of the literature, DL frameworks are employed in various manners within the field of cardiac electrophysiology. Several studies focus on the PDE problem derived from electrical wave propagation in cardiac tissue. In these studies, the DL framework is generally trained by incorporating geometrical information about the domain and physical constraints. PINNs represent a widely utilized approach for addressing electrophysiology problems by embedding physical laws inside the loss function. However, in the PINNs and other frameworks, when geometry and unique physical constraints are integrated within the model, training the same model repeatedly for different physical conditions might be required. Thus, the training phase of the model is likely to create a bottleneck and limit the clinical applicability. More complex approaches demand sophisticated and complex algorithms to deal with this problem. However, the generalization of these algorithms has yet to be fully established. Also, the accuracy of these methodologies often diminishes with simulation time. Another focus is predicting transmembrane potential at the cellular level as an ODE problem. Training a DL-based model at the cellular level is simpler than teaching the dynamics of the whole domain, and these models can be less computationally expensive and faster to train. The current body of research addressing this problem at the ODE scale remains very limited. This study proposes a DL model for the prediction of action potential values at the cellular level. There is also a gap in the literature for combining data-driven DL models with conventional numerical methods. The finite element method (FEM) is an important, well-established, and common numerical approach for solving PDEs. An important novelty of the proposed approach in this thesis is that it can successfully integrate the DL framework with the finite el-

ement method. Therefore, the DL framework can bridge the gap between the ODE problem at the cellular level and the PDE problem of the considered domain. This approach can be extended to other conventional numerical methods as well. The well-understood strength of the finite element method ensures the reliability of the solution and physical consistency, while the DL model improves the time efficiency. Another advantage is the accuracy of the proposed model does not diminish over time. The behavior of the DL model can be modified in the desired direction by adjusting the distribution of training data; hence, the flexibility of the proposed framework can be enhanced for the considered problem. The framework presented in this thesis can be rapidly modified and extended to solve different kinds of problems.

1.4 Aim and Scope of Thesis

The aim of this thesis is to develop a computational approach that combines a robust numerical method with a DL-based algorithm for solving cardiac electrophysiology problems. In this contribution, computational gain, accuracy, and consistency over long simulation periods of the proposed framework are investigated. A specific DL architecture is designed for the prediction of action potential values at the cellular level. As a first step, the DL model is developed to learn from a high-fidelity model of electrophysiology. The chosen high fidelity model in this thesis is the ten Tussher-Panfilov model [13]. Then, the DL model is trained with the solution of the ten Tusscher-Panfilov model under changing conditions. These conditions include varying time frequency of exciting stimuli and magnitude of the given stimuli. The most important part of the training is predicting the potential values with high accuracy over a long time period and capturing the complex phenomena of the ten Tusscher-Panfilov model. The external stimuli are an input for the DL model along with the past action potential values. The proposed framework makes us capable of solving PDEs by creating a correlation between external stimuli and the conductivity term of the PDE. FEM is used to solve the PDE over the domain, while DL is predicted electrical response at the cellular level. A lumping technique is employed within the FEM to enhance the framework's performance and to increase the correlation between the DL model and the FEM.

Several tests are run with the proposed framework to demonstrate the performance and applicability in more complex conditions. The performance of the proposed framework is compared with and without the lumping technique applied to the FEM solver. Also, the framework's performance is demonstrated with different time step sizes, and wave propagation is observed in planar and nonplanar conditions. A realistic ventricular geometry in the two dimensional domain of the heart is used to demonstrate the capabilities of the framework in a realistic multidimensional setting.

1.5 Outline

The rest of the thesis is organized as follows: in Chapter 2, the anatomical structure of the human heart is explained briefly. It is followed by an introduction to the electrophysiological aspects and behaviors of the heart. The ground truth model used for the simulation of electrophysiology in this study is introduced and its formulation is explained.

Chapter 3 summarizes the finite element method (FEM) and lumping technique. A FEM formulation based on the problem at hand is explained step by step for unfamiliar readers.

In Chapter 4, DL is briefly introduced along with the necessary terms. DL layers and modeling hyperparameters used in this study are described, and their purpose and contribution to the model are explained.

Chapter 5 gives the chosen architecture for the DL model. The chosen way for obtaining the necessary data and the preprocessing steps for the data are explained in detail. The outcome of the training phase and the evolution of loss functions are demonstrated. The accuracy of the trained model is demonstrated under different conditions, and computation times are presented to the reader.

In Chapter 6, the trained DL model is extended to work in conjunction with the FEM, which is responsible for solving the PDE. This extension process and the structure of the algorithm are explained in detail.

Chapter 7 provides the results of the employed framework. Multiple tests under dif-

ferent conditions are conducted, and the results of these tests are presented and evaluated. These tests include the effects of the lumping, time step size, and the excitation procedure. Also, the framework is tested for anatomically realistic geometry. Finally, a test for capturing the complex reentry mechanism is conducted.

In Chapter 8, the results of the created framework are evaluated. The advantages of the frameworks are discussed, and weaker sides are pointed out. Finally, the directions for future studies are shared with the reader.

CHAPTER 2

CARDIAC ELECTROPHYSIOLOGY

2.1 Anatomy of the Heart

The heart is a specialized muscular organ that undertakes the rhythmic contraction process, facilitating the blood flow from the low-pressure venous side to the high-pressure arterial side of the circulation. Pumping of the blood occurs due to the sequential contraction of the different heart chambers and the presence of the valves within itself. A summary of the cardiac cycle and important anatomical components of the heart is given in this section. The human heart consists of four chambers: the right ventricle, left ventricle, right atrium, and left atrium.

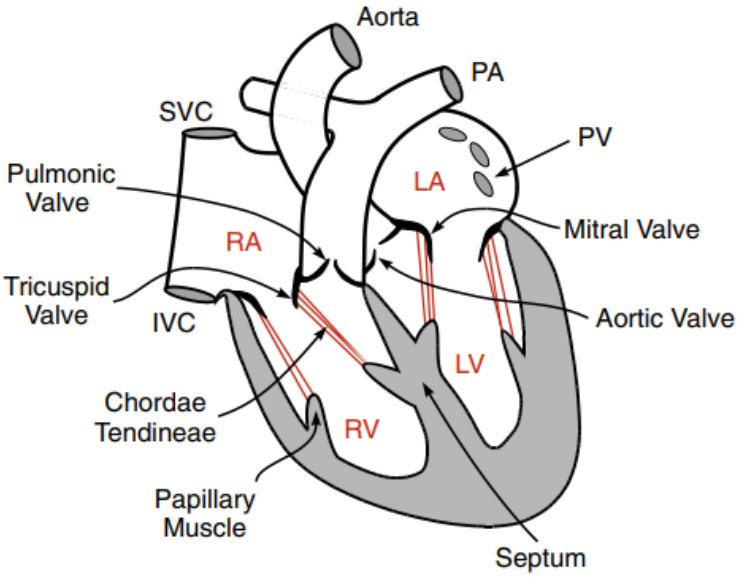

Figure 2.1: Anatomical representation of the heart [2].

The anatomical structure of the heart is depicted in Figure 2.1. The right atrium re-

ceives the blood returning from systemic circulation. Blood in the right atrium flows across the tricuspid valve into the right ventricle. The right ventricle pumps blood to the pulmonary artery, which is separated from the right ventricle with the pulmonic valve. The pulmonary artery carries the blood to the lungs, where the oxygen concentration increases. Oxygen-rich blood returns to the heart by four pulmonary veins. Oxygen-rich blood gets collected in the left atrium first. Blood then flows from the left atrium to the left ventricle across the mitral valve. The left ventricle has a thicker muscular wall than the right ventricle, which can generate enough pressure for blood to reach distant parts of the body. The contraction of the left ventricle pumps the blood across the aortic wave into the aorta, and blood is distributed across the body through the aorta. The right and left atria are divided by interatrial septum, while the left and right ventricles are divided by thicker interventricular septum. Another essential anatomical feature of the heart is the fibrous strands on leaflets of tricuspid and mitral waves. These fibrous strands attach to papillary muscles located on the ventricular walls. During the ventricular contraction, tension is generated by papillary muscles, which prevents valves from bulging back and leaking blood into the atria.

2.2 Electrophysiology of the Heart

Rhythmic contraction and relaxation of the heart are necessary to fulfill their fundamental role in pumping blood. The electrical system within the heart functions as a control mechanism for the contraction-relaxation process. Specialized structures that are unique to the heart generate and conduct electrical impulses. Generation and conduction of electrical impulses are achieved by changes in the state of ion channels of cell membranes, the concentration of different ions across the cell membrane, and the membrane permeability to these ions. The concentration of K^+, Na^+, and Ca^{++} ions are the most important factors for determining potential across the cell membrane [2].

Cardiac cells can be classified into two primary groups based on their capability for generating electrical activity. Pacemaker cells are highly specialized cells that can depolarize rhythmically and initiate an action potential cycle. The propagation of depolarizing currents from adjacent cells triggers the action potential of non-pacemaker cells. The action potential behavior of cardiac cells differs significantly from other

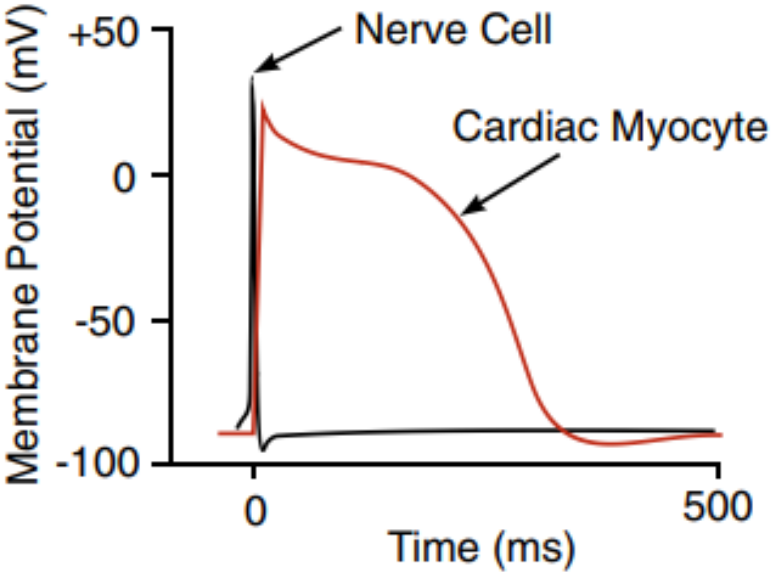

Figure 2.2: Action potential comparison between a nerve cell and a nonpacemaker cardiac myocyte [2].

types of cells within the human body. Figure 2.2 compares the action potential of a nonpacemaker cardiac cell with a nerve cell. Action potential duration is approximately 1 to 2 milliseconds for a nerve cell and 200 to 400 milliseconds for a cardiac myocyte. The difference proves the unique structure of the heart myocytes.

Nonpacemaker cells, such as atrial and ventricular myocytes and Purkinje fiber cells, are responsible for the conduction of the electrical currents across the heart. Purkinje fibers are composed of electrically excitable cells that can transmit action potential faster than other cell types in the cardiac tissue. A cardiac myocyte is a contractable and excitable cell that is intrinsic to the muscle structure of the heart. Electrical changes within the myocytes initiate the contraction. The contraction of cardiomyocytes generates the necessary force to pump the blood out of the heart. Figure 2.3 demonstrates the action potential stages of a ventricular myocyte and changes in the ionic conductance across the cell membrane with respect to the conductivity of important ions. Phase 4 is the resting potential for a myocyte. Without any external stimuli, the potential of nonpacemaker cells stays close to the resting potential. When these cells rapidly depolarize above a threshold potential, the conductance of voltage-

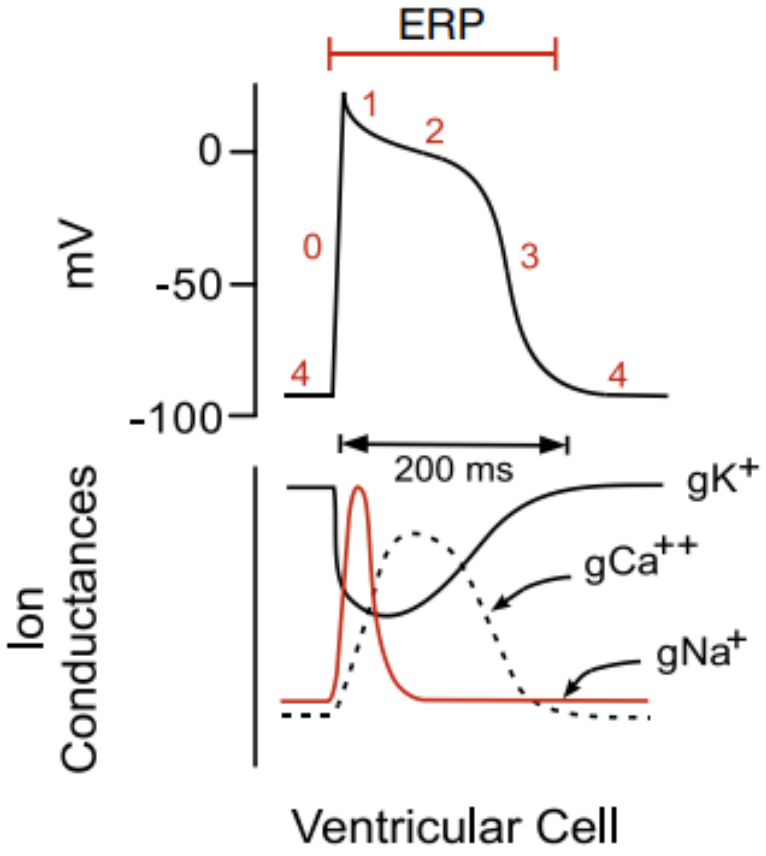

Figure 2.3: Changes of ion conductances with the action potential of a nonpacemaker cell [2].

gated fast Na^+ channels increases, and K^+ conductance decreases very rapidly. As a result of this process, Phase 0 starts. Activation of a special type of K^+ channels starts an initial repolarization named Phase 1. An increase in slow inward Ca^{++} conductance delays the repolarization, and then the potential value of the cell enters a plateau phase (Phase 2). Finally, the conductance of K^+ and Ca^{++} starts to return to the resting values, and repolarization (Phase 3) occurs. A critical characteristic of the action potential is that the cell is inexcitable during phases 0,1,2 and some parts of 3. The inexcitable period is called the effective refractory period.

The main group of pacemaker cells is within the sinoatrial node, which is located in the right atrium. Specialized cardiomyocytes in the sinoatrial node set the rhythm for cardiac contraction by repeatedly generating action potentials. Since pacemaker cells keep generating action potentials, they do not have a true resting state. Figure 2.4

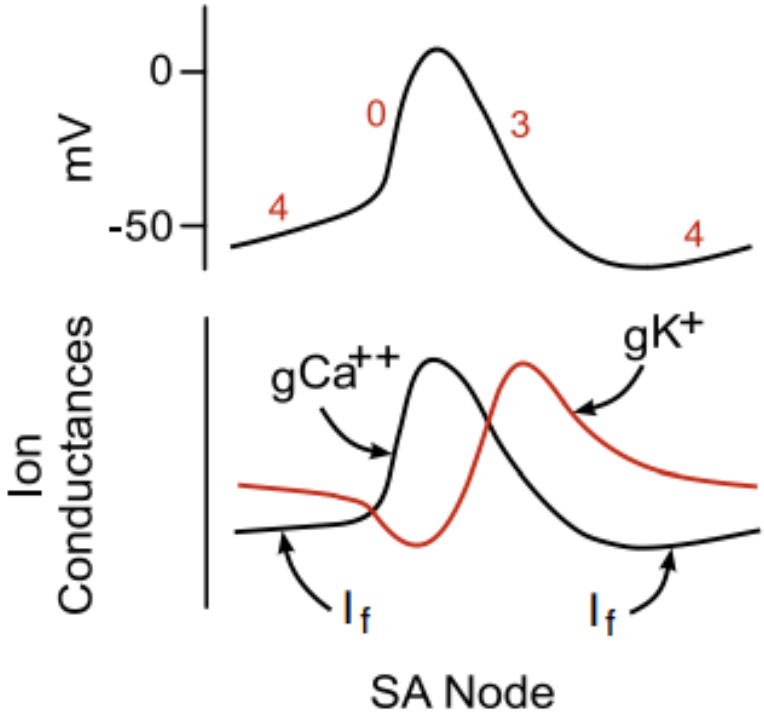

Figure 2.4: Changes of ion conductances with the action potential of a pacemaker cell [2].

demonstrates the action potential stages of a pacemaker cell and ion conductance for a pacemaker cell membrane. In these cells, depolarization occurs due to increased Ca^{++} conductivity (Phase 0). Depolarization response is much slower compared to that of nonpacemaker cells. Phase 0 triggers voltage-dependent K^{+} channels to open, and in return, repolarization starts (Phase 3). After the completion of Phase 3, the internal mechanisms of the cell increase the potential back to the depolarization threshold (Phase 0) and complete the cycle of the pacemaker cell.

Different factors can affect the electrophysiological behavior of the heart. The nervous system regulates the sinoatrial node behavior and conduction of action potential within the heart. Hormones, drugs, and environmental factors can modify the sinoatrial node's firing rate and tissue conductivity.

Disruptions in the electrophysiological architecture of the heart can perturb the mechanical properties and potentially lead to significant health issues. Various electro-

physiological disorders exist, and accurate diagnosis of these problems holds great importance. The ECG is the most common and significant diagnostic tool for detecting problems regarding heart electrophysiology. An ECG test is performed by attaching sensors to specific locations on the patient's body. A machine records the electrical signals generated by the heart, and then the electrical activity of the heart can be evaluated by a healthcare specialist. Infarction, different kinds of arrhythmias, and problems in the conduction system can be detected by ECG.

2.3 Mathematical Models of Cardiac Electrophysiology

Mathematical modeling of the action potential of cardiac cells can provide significant advantages in specialized drug development, understanding heart behaviors, and rapidly establishing a diagnosis for heart diseases. One of the first mathematical formulations of the electrophysiological behavior of a cell was proposed by Alan Hodgkin and Andrew Huxley in 1952 [15]. The proposed model explains the intricate ionic mechanisms that govern the initiation and conduction of action potentials in the squid giant axon. In more recent years, various mathematical models have been proposed for cardiac electrophysiology, for instance, [14, 66]. Electrophysiology models can be classified as phenomenological models and ionic models. Phenomenological models are simplified representations of observed phenomena based on empirical data or observed patterns. Phenomenological models are commonly utilized to understand underlying mechanisms that are highly complex or not discovered fully. Ionic models are mathematical representations that describe the behavior of the cell membrane gates and the movement of ions. Ionic models also involve physically meaningful parameters such as ion concentration and cell membrane conductivity to specific ions. These models can capture the complex interactions between ions, ion channels, and cell membranes and can be used to describe specific cell types of regions within the heart.

Some common phenomenological model examples include Fitzhugh-Nagumo model, Aliev-Panfilov model, and Mitchell-Schaffer model [16, 67, 68, 69]. While Fitzhugh-Nagumo model is generally used to simulate pacemaker cells, Aliev-Panfilov and Mitchell-Schaffer models are used for cardiac myocytes. On the other hand, Luo-

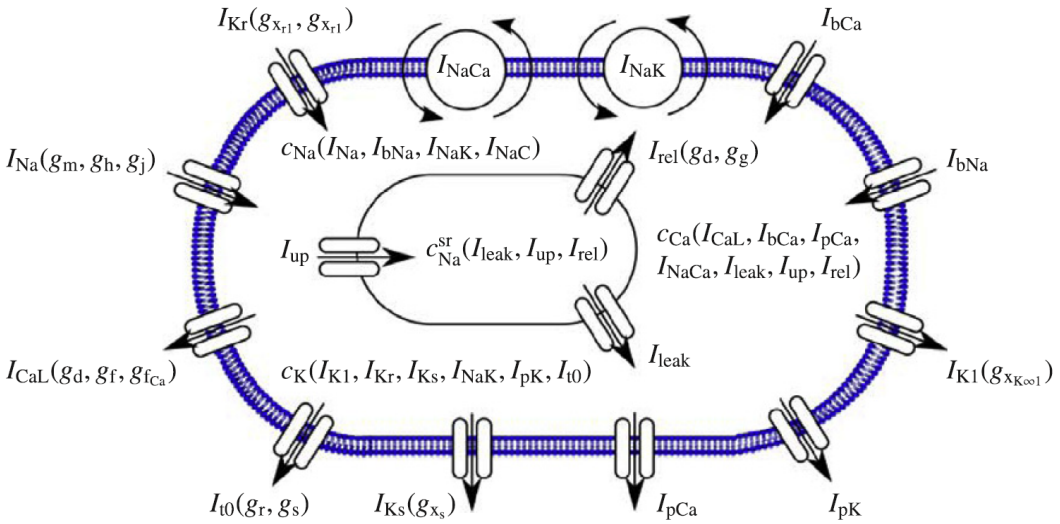

Figure 2.5: ten Tusscher-Panfilov model of a human ventricular cardiomyocyte [1].

Rudy, Courtemanche-Ramirez-Nattel, and ten Tusscher-Panfilov models can be given as examples for ionic models [14, 66, 13, 70]. While Courtemanche-Ramirez-Nattel model is designed to represent the action potential of atrial cells, Luo-Rudy and ten Tusscher-Panfilov models are designed for ventricular myocytes. Phenomenological models have faster performance than ionic models in general due to a lower number of variables. However, ionic models give more physically meaningful insight to researchers and higher accuracy. Details of ionic models can vary a lot, and the number of variables in ionic models can go up to 60. In this thesis, the ten Tusscher-Panfilov model is chosen to analyze electrophysiological behavior since this model can reflect the most important characteristics of the action potential and provide high accuracy for the cost of an increase in the computation time. Figure 2.5 illustrates ten Tusscher-Panflov model. The model is characterized by ion concentrations, $\mathbf{c}_{ion} = [c_{Na}, c_K, c_{Ca}, c^{sr}_{Ca}]^T$, ionic currents, $\mathbf{I}_{crt} = [I_{Na}, I_{bNa}, I_{NaK}, I_{NaCa}, I_{K1}$,-$I_{Kr}, I_{Ks}, I_{pK}, I_{t0}, I_{CaL}, I_{bCa}, I_{pCa}, I_{leak}, I_{up}, I_{rel}]^T$ and gating variables, $\mathbf{g}_{gate} = [g_m$,-$g_h, g_j, g_{xr1}, g_{xr2}, g_{xs}, g_r, g_s, g_d, g_f, g_{xK11\infty}, g_{fCa}, g_g]^T$. The model considers $n_{ion} = 4$ ion concentrations, $n_{crt} = 15$ ionic currents and $n_{gate} = 13$ gating variables. The following parts will formulate the constitutive equations for human ventricular myocytes.

2.3.1 Constitutive Equations of ten Tusscher-Panfilov Model

The ten Tusscher-Panfilov model considers ion concentrations, ionic currents, and gating variables. The model considers $n_{ion} = 4$ ion concentrations defined by,

$$\dot{\mathbf{c}}_{ion} = \dot{\mathbf{c}}_{ion}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion}) \quad \text{with} \quad \mathbf{c}_{ion} := [c_{Na}, c_K, c_{Ca}, c_{Ca}^{sr}], \tag{2.1}$$

where c_{Na}, c_K, and c_{Na} are the intracellular sodium, potassium, and calcium concentration, respectively, and c_{Ca}^{sr} is the calcium concentration of sarcoplasmic reticulum. The transmembrane potential is denoted with ϕ. The $\dot{()}$ symbol used over a variable demonstrates the derivative with respect to time of that variable. Ionic currents of the model are,

$$\begin{aligned}
\mathbf{I}_{crt} =& \mathbf{I}_{crt}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion}) \quad \text{with} \\
\mathbf{I}_{crt} =& [I_{Na}, I_{bNa}, I_{NaK}, I_{NaCa}, I_{K1}, I_{Kr}, I_{Ks}, I_{pK}, I_{t0}, I_{CaL}, I_{bCa}, I_{pCa}, \\
& I_{Leak}, I_{up}, I_{rel}]^T .
\end{aligned} \tag{2.2}$$

Sodium related currents are given as I_{Na}, I_{bNa}, I_{NaK}, and I_{NaCa}, and they induce changes on the intracellular sodium concentration c_{Na}. Potassium related currents are, I_{K1}, I_{Kr}, I_{Ks}, I_{NaK}, I_{pK}, and I_{t0}. These currents induce changes in the intracellular potassium concentration c_K. Calsium related currents, I_{CaL}, I_{bCa}, I_{pCa}, I_{NaCa}, I_{leak}, I_{up}, and I_{rel} induce changes in the intracellular calsium concentration c_{Ca}. Finally, the currents, I_{leak}, I_{up}, and I_{rel} lead to changes in the calcium concentration in the sarcoplasmic reticulum c_{Ca}^{sr}. Channel states of the cell are associated with $n_{gate} = 13$ gating variables. The gating variables of the model are,

$$\begin{aligned}
&\dot{\mathbf{g}}_{gate}^{I} = \dot{\mathbf{g}}_{gate}^{I}(\phi, \mathbf{g}_{gate}^{I}), \\
&\dot{\mathbf{g}}_{gate}^{II} = \dot{\mathbf{g}}_{gate}^{II}(\phi, \mathbf{g}_{gate}^{II}, \mathbf{c}_{ion}), \\
&\text{with} \\
&\mathbf{g}_{gate}^{I} = [g_m, g_h, g_j, g_{xr1}, g_{xr2}, g_{xs}, g_r, g_s, g_d, g_f]^T, \\
&\mathbf{g}_{gate}^{II} = [g_{xK1\infty}, g_{fCa}, g_g]^T.
\end{aligned} \tag{2.3}$$

In Equation (2.3), g_m, g_h, g_j control the fast sodium channel, $g_{K1\infty}$ controls the inward rectifier channel, g_{xr1}, g_{xr2} control the rapid delayed rectifier channel, g_{xs} controls the slow delayed rectifier channel, g_r, g_s control the transient outward channel, g_d, g_f, g_{fCa} control the L-type calcium channel and g_d, g_g control the sarcoplasmic reticulum calcium release channel. For each ion, the classical Nernst equation,

$$\phi_{ion} = \frac{RT}{z_{ion}F} \log \frac{c_{ion0}}{c_{ion}} \quad \text{with} \quad \phi_{ion} = [\phi_{Na}, \phi_K, \phi_{Ca}]^T \tag{2.4}$$

can be used to calculate Nernst or reversal potential ϕ_{ion}. The z_{ion} is the elementary charge per ion, $z_{ion} = 1$ for singly-charged ions, and $z_{ion} = 2$ for double-charged ions. In Equation (2.4) R is the gas constant, F is the Faraday constant, and T is the temperature. The rest of the material parameters utilized in this model are given in Table 2.2. The following part of this chapter specifies the ion concentrations, currents, and gating variables. Currents calculated from this model define the source term f^ϕ as follows,

$$f^\phi = -[I_{Na} + I_{bNa} + I_{NaK} + I_{NaCa} + I_{K1} + I_{Kr} + I_{Ks} + I_{pK} + I_{t0} + I_{CaL} + I_{bCa} + I_{pCa}]. \tag{2.5}$$

Table 2.1: Utilized physical units in this study.

Physical Quantity	Unit Name	Abbreviation
Time	milliseconds	ms
Voltage	millivolts	mV
Ionic Currents	picoamperes per picofarad	pA/pF
Ionic Currents Across the Membrane of the Sarcoplasmic Reticulum	millimolar per millisecond	mM/ms
conductances	nanosiemens per picofarad	nS/pF
Intracellular and Extracellular Ion Concentrations	millimoles per liter	mmol/L

Physical units used in the rest of this study are demonstrated in Table 2.1. Next, the evolution of some parameters is explained in detail for the sake of completeness.

2.3.1.1 Formulations of Sodium Related Variables

In the model, the evolution of the sodium concentration is calculated with

$$\dot{c}_{Na} = -\frac{C}{VF}[I_{Na} + I_{bNa} + 3I_{NaK} + 3I_{NaCa}], \tag{2.6}$$

where fast sodium current I_{Na}, the background sodium current I_{bNa}, the sodium potassium pump current I_{NaK}, and sodium calcium exchanger I_{NaCa} current are scaled by the membrane capacitance per unit surface area C, the cytoplasmic volume V, and the Faraday constant F. The sodium-related currents are defined as follows,

$$\begin{aligned}
I_{Na} =& C_{Na}^{Max} g_m^3 g_h g_j [\phi - \phi_{Na}], \\
I_{bNa} =& C_{bNa}^{Max} [\phi - \phi_{Na}], \\
I_{NaK} =& I_{NaK}^{Max} [c_{K0} c_{Na}][c_{Na} + c_{NaK}][c_{K0} + c_{KNa}] \\
& \times [1 + 0.1245 e^{-0.1\phi \frac{F}{RT}} + 0.0353 e^{\phi \frac{F}{RT}}]^{-1}, \\
I_{NaCa} =& I_{NaCa}^{Max} [e^{\gamma \phi \frac{F}{RT}} c_{Na}^3 c_{Ca0} - e^{[\gamma - 1]\phi \frac{F}{RT}} c_{Na0}^3 c_{Ca} \gamma_{NaCa}] \\
& \times [[c_{NaCa}^3 + c_{Na0}^3][c_{CaNa} + c_{Ca0}][1 + k_{NaCa}^{sat} e^{[\gamma - 1]\phi \frac{F}{RT}}]]^{-1},
\end{aligned} \tag{2.7}$$

where the scaling factors are the maximum fast sodium conductance C_{Na}^{max}, the maximum background sodium conductance C_{bNa}^{max}, the maximum sodium potassium pump current I_{NaK}^{max}, and the maximum sodium calcium exchanger current I_{NaCa}^{max}. Additional parameters in the equations for sodium potassium pump current I_{NaK} and for the sodium calcium exchanger current I_{NaCa} are extracellular sodium, potassium, and calcium concentrations C_{NaO}, C_{KO}, C_{CaO}, the half saturation constants c_{NaCa}, c_{CaNa}, c_{KNa}, c_{NaK}, the sodium calcium saturation factor k_{NaCa}^{sat}, the outward sodium calcium pump current enhancing factor γ_{NaCa}, and the voltage dependent sodium parameter c. The fast sodium current I^{Na} is characterized through a three-gate formulation with τ term to reflect the time constant associated with the steady state. The sodium activation gate takes the form

$$\begin{aligned}
g_m^{\infty} &= [1 + e^{[-56.85 - \phi]/9.03}]^{-2}, \\
\tau_m &= 0.1[1 + e^{[-60 - \phi]/5}]^{-1}][[1 + e^{[35 + \phi]/5}]^{-1}] + [1 + e^{[\phi - 50]/200}]^{-1}]].
\end{aligned} \tag{2.8}$$

The fast sodium inactivation gate is defined by

$$
\begin{aligned}
g_h^{\infty} &= [1 + e^{[\phi+71.55]/7.43}]^{-2}, \\
\tau_h &= \begin{cases} 0.1688[1 + e^{-[\phi+10.66]/11.1}], & \text{if} \quad \phi \geq -40, \\ [0.057e^{-[\phi+80]/6.8} + 2.7e^{0.079\phi} + 3.1 \times 10^5 e^{0.3485\phi}]^{-1}, & \text{if} \quad \phi < -40. \end{cases}
\end{aligned} \tag{2.9}
$$

The slow sodium inactivation gate is defined by

$$
\begin{aligned}
g_j^{\infty} &= [1 + e^{[\phi+71.55]/7.43}]^{-2}, \\
\tau_j &= [\alpha_j + \beta_j]^{-1}, \\
\alpha_j &= \begin{cases} 0, & \text{if} \quad \phi \geq -40, \\ [-2.5428 \times 10^4 e^{0.2444\phi} - 6.948 \times 10^{-6} e^{-0.04391\phi}] \\ [\phi + 37.78][1 + e^{0.311[\phi+79.23]}]^{-1}, & \text{if} \quad \phi < -40, \end{cases} \\
\beta_j &= \begin{cases} 0.6e^{0.057\phi}[1 + e^{-0.1[\phi+32]}]^{-1}, & \text{if} \quad \phi \geq -40, \\ 0.02424e^{-0.01052\phi}[1 + e^{-0.1387[\phi+40.14]}]^{-1}, & \text{if} \quad \phi < -40. \end{cases}
\end{aligned} \tag{2.10}
$$

2.3.1.2 Formulations of Potassium Related Variables

The evolution of the potassium concentration

$$
\dot{c}_K = -\frac{C}{VF}[I_{K1} + I_{Kr} + I_{Ks} - 2I_{NaK} + I_{pK} + I_{stim}] \tag{2.11}
$$

is calculated by six different currents. Inward rectifier current I_{K1}, the rapid delayed rectifier current I_{Kr}, slow delayed rectifier current I_{Ks}, the transient outward current, sodium potassium pump current I_{K1}, the plateau potassium current I_{pK}, and the external stimulus current I_{stim} govern the evolution of potassium concentrations and are defined as follow,

$$
\begin{aligned}
I_{K1} =& C_{K1}^{max} g_{K1}^{\infty} [c_{K0}/5.4]^{1/2} [\phi - \phi_K], \\
I_{Kr} =& C_{Kr}^{max} g_{xr1} g_{xr2} [c_{K0}/5.4]^{1/2} [\phi - \phi_K], \\
I_{Ks} =& C_{Ks}^{max} g_{xs}^2 g_{xr2} [\phi - \phi_K], \\
I_{NaK} =& I_{NaK}^{max} [c_{K0} c_{Na}] [[C_{Na} + c_{NaK}][c_{K0} + c_{KNa}] \\
& \times [1 + 0.1245 e^{-0.1\phi F/RT} + 0.0.0353 e^{[-\phi F/RT]}]]^{-1}, \\
I_{pK} =& C_{pK}^{max} [1 + e^{[25-\phi]/5.98}]^{-1} [\phi - \phi_K], \\
I_{t0} =& C_{t0}^{max} g_r g_s [\phi - \phi_K].
\end{aligned}
\tag{2.12}
$$

Scaling factors for these currents are the maximum inward rectifier conductance C_{K1}^{max}, the maximum rapid delayed rectifier conductance C_{Kr}^{max}, the maximum slow delayed rectifier conductance for epicardial $C_{Ks,epi}^{max}$, endocardial $C_{Ks,endo}^{max}$, and M $C_{Ks,M}^{max}$ cells, the maximum sodium potassium pump current I_{NaK}^{max}, the maximum potassium pump conductance C_{pK}^{max}, and the maximum transient outward conductance for epicardial $C_{t0,epi}^{max}$, endocardial $C_{t0,endo}^{max}$, and M $C_{t0,M}^{max}$ cells. The time-independent inward rectification factor g_{K1}^{∞} has a significant role in the calculation of maximum inward rectifier current I_{K1} and is defined as,

$$
\begin{aligned}
& g_{K1}^{\infty} = \alpha_{K1} [\alpha_{K1} + \beta_{K1}]^{-1}, \\
& \text{with} \\
& \alpha_{K1} = 0.1 [1 + e^{0.06[\phi - \phi_k - 200]}]^{-1}, \\
& \beta_{K1} = [3 e^{0.0002[\phi - \phi_k + 100]} + e^{0.1[\phi - \phi_k - 10]}][1 + e^{-0.5[\phi - \phi_k]}]^{-1}.
\end{aligned}
\tag{2.13}
$$

Steady state value and time constant for the activation gate of the rapid delayed rectifier current I_{Kr} are given as

$$
\begin{aligned}
g_{xr1}^{\infty} &= [1 + e^{[-26-\phi]/7}]^{-1}, \\
\tau_{xr1} &= 2700 [1 + e^{[-45-\phi]/10}]^{-1} [1 + e^{[30+\phi]/11.5}]^{-1},
\end{aligned}
\tag{2.14}
$$

and steady state value and time constant for the inactivation gate are given as

$$
\begin{aligned}
g_{xr2}^{\infty} &= [1 + e^{[88+\phi]/24}]^{-1}, \\
\tau_{xr2} &= 3.36[1 + e^{[-60-\phi]/20}]^{-1}[1 + e^{[\phi-60]/20}]^{-1}.
\end{aligned}
\tag{2.15}
$$

The activation gate for the delayed rectifier current I_{Ks} is parameterized as follows

$$
\begin{aligned}
g_{xs}^{\infty} &= [1 + e^{[-5-\phi]/14}]^{-1}, \\
\tau_{xs} &= 1100[1 + e^{[-10-\phi]/6}]^{-1/2}[1 + e^{[\phi-60]/20}]^{-1}.
\end{aligned}
\tag{2.16}
$$

The activation gate for the transient potassium outward current I_{Ks} is parameterized as follows

$$
\begin{aligned}
g_{r}^{\infty} &= [1 + e^{[20-\phi]/6}]^{-1}, \\
\tau_{r} &= 9.5[1 + e^{-[40+\phi]^2/1800}] + 0.8.
\end{aligned}
\tag{2.17}
$$

The voltage dependent potassium inactivation gate exhibits different kinds of behavior for epicardial and endocardial cells. The parameterization of this gate is given as

$$
\begin{aligned}
epicardium &\begin{cases} g_{s}^{\infty} = [1 + e^{[20+\phi]/5}], \\ \tau_{s} = 85e^{-[45+\phi]^2/320} + 5[1 + e^{[-20+\phi]/5}] + 3, \end{cases} \\
endocardium &\begin{cases} g_{s}^{\infty} = [1 + e^{[28+\phi]/5}], \\ \tau_{s} = 1000e^{-[67+\phi]^2/1000} + 8. \end{cases}
\end{aligned}
\tag{2.18}
$$

Introduced half saturation constants in potassium related equations are c_{KNa} and c_{NaK}.

2.3.1.3 Formulations of Calcium Related Variables

Calcium is a critical ion in cardiac tissue since it is an important factor in electrophysiological behavior and directly impacts the contraction mechanism. The evolution of the calcium concentration is given as

$$\dot{c}_{Ca} = \gamma_{Ca}[-\frac{C}{2VF}[I_{CaL} + I_{bCa} + I_{pCa} - 2I_{NaCa}] + I_{leak} - I_{up} + I_{rel}]. \quad (2.19)$$

Intracellular calcium concentration is affected by L-type calcium current I_{CaL}, the background calcium current I_{bCa}, the plateau calcium current I_{pCa}, the sodium calcium pump current I_{NaCa}, the leakage current I_{leak}, the sarcoplasmic reticulum uptake current I_{up}, and the sarcoplasmic reticulum release current I_{rel}. These currents are defined in the model as follows,

$$\begin{aligned}
I_{CaL} =& C_{CaL}^{max} g_d g_f g_{fCa} [4\phi F^2]/[RT][c_{Ca} e^{2\phi F/[RT]} - 0.341 C_{Ca0}][e^{2\phi F/[RT]} - 1]^{-1},\\
I_{bCa} =& C_{bCa}^{max}[\phi - \phi_{Ca}],\\
I_{pCa} =& C_{pCa}^{max} c_{Ca}[c_{pCa} + c_{Ca}]^{-1},\\
I_{NaCa} =& I_{NaCa}^{max} c_{Ca}[e^{\gamma\phi F/[RT]} c_{Na}^3 c_{Ca0} - e^{[\gamma-1]\phi F/[RT]} c_{Na0}^3 c_{Ca} \gamma_{NaCa}\\
& \times [[c_{NaCa}^3 + c_{Na0}^3][c_{CaNa} + c_{Ca0}][1 + k_{NaCa}^{sat} e^{[\gamma-1]\phi F/RT}]]^{-1},\\
I_{leak} =& I_{leak}^{max}[c_{Ca}^{sr} - c_{Ca}],\\
I_{up} =& I_{up}^{max}[1 + c_{up}^2/c_{Ca}^2]^{-1},\\
I_{rel} =& I_{rel}^{max} g_d g_g [1 + \gamma_{rel} c_{Ca}^{sr2}[c_{rel}^2/c_{Ca}^{sr2}]^{-1}].
\end{aligned} \quad (2.20)$$

The scaling factors are the maximum calcium conductance C_{CaL}^{max}, the maximum background calcium conductance C_{bCa}^{max}, the maximum plateau calcium conductance C_{pCa}^{max}, the maximum sodium calcium pump current I_{NaCa}^{max}, the maximum leakage current I_{leak}^{max}, maximum sarcoplasmic reticulum calcium uptake current I_{up}^{max}, and the maximum sarcoplasmic reticulum calcium release current I_{rel}^{max}. The long-lasting L-type calcium channel is controlled by the voltage-dependent activation gate that has the following formulation

$$\begin{aligned}
g_d^{\infty} &= [1 + e^{[-5-\phi]/7.5}]^{-1},\\
\tau_d &= [1.4[1 + e^{[-35-\phi]/13}]^{-1} + 0.25][1.4[1 + e^{[5+\phi]/5}]] + [1 + e^{[50-\phi]/20}].
\end{aligned} \quad (2.21)$$

The formulation for the voltage-dependent inactive gate is

$$
\begin{aligned}
g_f^{\infty} &= [1 + e^{[20+\phi]/7}]^{-1}, \\
\tau_f &= 1125e^{-[\phi+27]^2/240} + 165[1 + e^{[25-\phi]/10}]^{-1} + 80.
\end{aligned}
\tag{2.22}
$$

Intracellular calcium dependent inactivation gate is characterized through

$$
\begin{aligned}
g_{fCa}^{\infty} &= 0.685[[1 + [C_{Ca}/0.000325]^8]^{-1} + 0.1[1 + e^{[c_{Ca}-0.0005]/0.0001}]^{-1} \\
&\quad + 0.2[1 + e^{[c_{Ca}-0.00075]/0.0008}]^{-1} + 0.23], \\
\tau_f &= \begin{cases} \infty, & \text{if} \quad g_{fCa}^{\infty} > g_{fCa} \quad \text{and} \quad \phi \geq 60 \quad \text{mV}, \\ 2 \quad \text{ms}, & \text{otherwise}. \end{cases}
\end{aligned}
\tag{2.23}
$$

Intracellular calcium dependent inactivation gate has two main states: no inactivation state and incomplete activation state. Finally, the calcium-induced calcium release current I_{rel} is characterized by the same gate activating the L-type calcium channel. Steady state and time constant formulation are given as,

$$
\begin{aligned}
g_g^{\infty} &= \begin{cases} [1 + c_{Ca}^6/0.00035^6]^{-1}, & \text{if} \quad C_{Ca} \leq 0.00035, \\ [1 + c_{Ca}^{16}/0.00035^{16}]^{-1}, & \text{otherwise}, \end{cases} \\
\tau_g &= \begin{cases} \infty, & \text{if} \quad g_g^{\infty} > g_g \quad \text{and} \quad \phi \geq -60 \quad \text{mV}, \\ 2 \quad \text{ms}, & \text{otherwise}. \end{cases}
\end{aligned}
\tag{2.24}
$$

The remaining parameters in these equations are half saturation constants for plateau calcium concentration c_{pCa}, the sarcoplasmic reticulum calcium uptake C_{up}, and the sarcoplasmic reticulum calcium release c_{rel}. The parameter γ_{NaCa} is introduced for regularization of the sodium calcium pump current I_{NaCa}. The parameter γ_{rel} is a weighting factor for the effect of the sarcoplasmic reticulum calcium concentration on sarcoplasmic reticulum calcium release I_{rel}. The total intracellular calcium concentration c_{cal}^{tot} is weighted by the γ_{Ca}

$$
\gamma_{Ca} = [1 + [c_{tot}c_{buf}][c_{Ca} + c_{buf}]^{-2}]^{-1}, \tag{2.25}
$$

where c_{tot} and c_{buf} correspond to the total and half saturation cytoplasmic calcium buffer concentrations, respectively.

2.3.1.4 Formulation of Sarcoplasmic Reticulum Calcium Related Variables

Evolution of the sarcoplasmic reticulum calcium concentration is defined as

$$\dot{c} + sr_{Ca} = \gamma_{Ca}^{sr} \frac{V}{V^{sr}} [-I_{leak} + I_{up} - I_{rel}]. \tag{2.26}$$

The sarcoplasmic reticulum calcium concentration is scaled by the ratio of the volume of the cytoplasm V to the volume of the sarcoplasmic reticulum V^{sr}. Contributing currents are formulated as,

$$\begin{aligned} I_{leak} &= I_{leak}^{max} [c_{Ca}^{sr} - c_{Ca}], \\ I_{up} &= I_{up}^{max} [1 + c_{up}^2 / c_{up}^2]^{-1}, \\ I_{rel} &= I_{rel}^{max} g_d g_g [1 + \gamma_{rel} c_{Ca}^{sr2} [c_{rel}^2 + c_{Ca}^{sr2}]^{-1}]. \end{aligned} \tag{2.27}$$

The scaling factors are the maximum leakage current I_{leak}^{max}, the maximum sarcoplasmic reticulum calcium uptake current I_{up}^{max}, and the maximum sarcoplasmic reticulum calcium release current I_{rel}^{max}. The half saturation constants are the calcium uptake c_{up} and the calcium release c_{rel}. Similar to the previous section, total calcium concentration in the sarcoplasmic reticulum is considered by summing free and buffered sarcoplasmic reticulum calcium concentrations. This sum is weighted by γ_{Ca}^{sr}

$$\gamma_{Ca}^{sr} = [1 + [c_{tot}^{sr} c_{buf}^{sr}][c_{Ca}^{sr} + c_{buf}^{sr}]^{-2}]^{-1}, \tag{2.28}$$

where c_{tot}^{sr} and c_{buf}^{sr} correspond to the total and half saturation sarcoplasmic reticulum calcium buffer concentrations, respectively.

Table 2.2: Material parameters of the ten Tusscher-Panfilov Model [1].

	Sodium Related	Potassium Related	Calcium Related	Calcium^{sr} Related
Concentrations	$c_{Na0} = 140$ mM	$c_{K0} = 5.4$ mM	$c_{Ca0} = 2$ mM	-
Maximum Currents	$I^{max}_{NaCa} = 1000$ pA/pF	$I^{max}_{NaCa} = 1000$ pA/pF		
	$I^{max}_{NaK} = 1.362$ pA/pF	$I^{max}_{NaK} = 1.362$ pA/pF	$I^{max}_{leak} = 0.08$ s^{-1}	$I^{max}_{leak} = 0.08$ s^{-1}
			$I^{max}_{up} = 0.425$ mM/s	$I^{max}_{up} = 0.425$ mM/s
			$I^{max}_{rel} = 8.232$ mM/s	$I^{max}_{rel} = 8.232$ mM/s
Maximum Conductances	$C^{max}_{Na} = 14.838$ nS/pF	$C^{max}_{K1} = 5.405$ nS/pF	$C^{max}_{CaL} = 0.175$ mm^3/[μFs]	
	$C^{max}_{bNa} = 0.00029$ nS/pF	$C^{max}_{Kr} = 0.0096$ nS/pF	$C^{max}_{bCa} = 0.000592$ nS/pF	
		$C^{max}_{Ks,epi} = 0.245$ nS/pF	$C^{max}_{pCa} = 0.825$ nS/pF	
		$C^{max}_{Ks,endo} = 0.245$ nS/pF		
		$C^{max}_{Ks,m} = 0.245$ nS/pF		
		$C^{max}_{pK} = 0.0146$ nS/pF		
		$C^{max}_{t0,epi} = 0.294$ nS/pF		
		$C^{max}_{t0,endo} = 0.073$ nS/pF		
		$C^{max}_{t0,m} = 0.294$ nS/pF		
Half Saturation Constants	$c_{CaNa} = 1.38$ mM		$c_{CaNa} = 1.38$ mM	
	$c_{NaCa} = 87.50$ mM		$c_{NaCa} = 87.50$ mM	
	$c_{KNa} = 1.00$ mM	$c_{KNa} = 1.00$ mM	$c_{pCa} = 0.0005$ mM	
	$c_{NaK} = 40.00$ mM	$c_{NaK} = 40.00$ mM	$c_{up} = 0.00025$ mM	$c_{up} = 0.00025$ mM
			$c_{rel} = 0.25$ mM	$c_{buf} = 0.25$ mM
			$c_{rel} = 0.001$ mM	$c^{sr}_{buf} = 0.3$ mM
Other Parameters	$ksat_{NaCa} = 0.10$	$p_{kNa} = 0.03$	$\gamma_{rel} = 2$	$\gamma_{rel} = 2$
	$\gamma_{NaCa} = 2.50$		$c_{tot} = 0.15$ mM	$c^{sr}_{tot} = 10$ mM
	$\gamma = 0.35$			
Gas Constant $R = 8.3143$ JK^{-1}mol^{-1}		Temperature $T = 310$ K		Cytoplasmic Volume $V = 16,404\ \gamma$m^3
Faraday Constant $F = 96.4867$ C/mmol		Membrane Capacitance $C = 185$ pF		Sarcoplasmic reticulum Volume $V^{sr} = 1094\ \gamma$m^3

CHAPTER 3

NUMERICAL METHODS

This chapter will introduce the numerical methods used to solve the PDE problem. The primary method utilized in this study is the finite element method. The finite element method and lumping technique, which is pivotal for the proposed algorithm, are explained in detail in the following sections.

3.1 Finite Element Method

The finite element method (FEM) is a mathematical method employed for the numerical approximation and analysis of complex physical problems; see, e.g., [1, 71]. It is a general and robust approach to real-world problems involving complex physics, geometries, and boundary conditions. In FEM, the given domain is considered a collection of subdomains, and these finite number of interconnected subdomains are called elements. In each element, the governing equation is approximated by a conventional variational method. The reason for seeking a solution via these elements is that it is easier to represent a complex function as a collection of simple polynomials [72]. It is crucial for these solutions over elements to fit with their neighbors and their derivatives up to a chosen order to be continuous at the connecting points.

There are three important steps to obtaining a solution via FEM. Discretizing complex domains reduces the problem to a collection of subdomains. Discretization can be a major advantage since complex geometries, such as the heart, get represented by elements with much simpler geometries. The second step is solving the problem for quantities of interest over every element by solving algebraic equations derived from governing equations. In the final step, solutions from every element are assem-

bled according to their relation. Then, the assembled system is solved for the whole domain, and interested quantities are calculated. The following section explains the FEM formulation for the electrophysiological problem studied in this thesis.

3.2 Finite Element Formulation of Electrophysiology

The utilization of the finite element method requires the mathematical definition of the problem intended to be solved. This study employs the finite element method to solve for transmembrane potential ϕ in a given physical domain. Finite element formulation for the electrophysiological problem is characterized through the transmembrane potential ϕ evolution in the spatiotemporal setting with

$$\dot{\phi} = \text{div}(\mathbf{q}(\phi)) + f^{\phi}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion}), \tag{3.1}$$

where the characterizing terms are flux $\mathbf{q}$ and the source f^{ϕ}, respectively. It is a common practice to express the potential flux and its derivative, respectively, in the following form

$$\mathbf{q} = \mathbf{D}\nabla\phi, \quad d_{\nabla\phi}\mathbf{q} = \mathbf{D}. \tag{3.2}$$

Here $\mathbf{D}$ is the conductivity tensor, which consists of isotropic and anisotropic parts. The conductivity tensor is defined as

$$\mathbf{D} = d^{iso}\mathbf{I} + d^{ani}\mathbf{n} \otimes \mathbf{n}, \tag{3.3}$$

where the diffusion tensor accounts for isotropic propagation d^{iso} and anisotropic propagation d^{ani} along preferred directions $\mathbf{n}$. The conductivity tensor's anisotropic part reflects the heart's structural anisotropy, such as myocardial fiber directions. Also, conductivities in the cardiac tissue can change significantly depending on the cell types. On the other hand, the source term in (3.1) is related to the cellular response at the micro level, and it can be described as

$$f^{\phi} = -\sum_{crt=1}^{ncrt} I_{crt}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion}). \tag{3.4}$$

The ionic currents are parameterized by gating variables $\mathbf{g}_{gate}$ and ion concentrations $\mathbf{c}_{ion}$. The ODE system for calculating the source term is explained in Chapter 2.3 in detail. In this problem, the transmembrane potential is considered the global degree of freedom in every finite element node. Gating variables and ion concentrations are stored locally at the integration points based on the formulations in [1]. The equation in (3.1) is discretized with finite elements in space and a finite difference approach in time. C^0-continuous finite element interpolation is chosen for the membrane potential. The expression in (3.1) is rewritten in a FEM applicable form as

$$R^{\phi} = \dot{\phi} - \text{div}(\mathbf{q}) - f^{\phi} = 0. \quad \text{in} \quad \mathcal{B}. \tag{3.5}$$

R^{ϕ} is the residual term. The boundary $\partial\mathcal{B}$ of the domain $\mathcal{B}$ can be divided to two parts as $\partial\mathcal{B}_q$ and $\partial\mathcal{B}_{\phi}$. Dirichlet boundary conditions are defined as $\phi = \overline{\phi}$ on $\partial\mathcal{B}_{\phi}$ while Neumann boundary conditions are defined for the flux vector as $\mathbf{q} \cdot \mathbf{n}$ on $\partial\mathcal{B}_q$. The vector $\mathbf{n}$ denotes the outward normal to $\partial\mathcal{B}$. In this electrophysiology problem, homogeneous Neumann boundary condition, $\mathbf{q} \cdot \mathbf{n} = 0$, is applied along the entire domain boundary $\partial\mathcal{B}$. The equations in (3.5) are integrated over the domain with the Neumann boundary conditions. The resulting equation is tested with a scalar-valued test function $\delta\phi$, and with the utilization of integration by parts and Gauss theorem, it is cast in the following weak form.

$$G^{\phi} = \int_{\mathcal{B}} \delta\phi\dot{\phi}dV + \int_{\mathcal{B}} \nabla\delta\phi \cdot \mathbf{q}dV - \int_{\delta\mathcal{B}_q} \delta\phi\overline{q}dA - \int_{\mathcal{B}} \delta\phi f^{\phi}dV = 0, \tag{3.6}$$

where $\overline{q}$ is the external flux applied to the boundary of the domain. Spatial discretization is performed on the domain of interest $\mathcal{B}$ by discretizing the domain into determined number of elements n_{el} with subdomains $\mathcal{B}^e$ as $\mathcal{B} = \bigcup_{e=1}^{n_{el}} \mathcal{B}^e$. Trial and test functions are interpolated with the same shape functions N in every element by following the isoparametric concept

$$\delta\phi^h|_{\mathcal{B}^e} = \sum_{i=1}^{n_{en}} N^i \delta\phi_i, \quad \phi^h|_{\mathcal{B}^e} = \sum_{j=1}^{n_{en}} N^j \phi_j, \tag{3.7}$$

where n_{en} is the number of elements and $\mathcal{B}^e$ is the domain for each element. The temporal discretization is performed by partitioning the time interval into subintervals of $[t_n, t_{n+1}]$. To solve for the potential at the time step t_{n+1}, the backward Euler time integration scheme is applied to the system.

$$\dot{\phi} = [\phi - \phi_n]/\Delta t, \tag{3.8}$$

where Δt is the time step. Equation (3.8) demonstrates the formulation of finite difference approximation. Subscript $(n+1)$ term is suppressed for clarity, and it will be omitted for the rest of the study. With the combination of temporal discretization and (3.6), the discrete residual can be written as follows,

$$R_I^\phi = \mathop{\mathbf{A}}_{e=1}^{n_{el}} \int_{\mathcal{B}^e} N^i \frac{\phi - \phi_n}{\Delta t} + \nabla N^i \cdot \mathbf{q} dV - \int_{\delta\mathcal{B}_q^e} N^i \overline{q} dA - \int_{\mathcal{B}^e} N^i f^\phi dV = 0. \tag{3.9}$$

The operator, $\mathbf{A}$, assembles all the element contributions from respective element nodes. The residual is highly nonlinear in the electrophysiology problem. Therefore, the iterative Newton-Raphson method is used in this framework. Using the Newton-Raphson method the converged solution can be obtained by repeatedly solving the consistently linearized system of equations. Apparently, the iterations are avoided if the system is composed of linear equations. Derivative of the residual term with respect to ϕ is needed to apply the Newton-Raphson method. Derivative of the residual term can be given as

$$K_{IJ}^\phi = d_{\phi_j} R_I^\phi = \mathop{\mathbf{A}}_{e=1}^{n_el} \int_{\mathcal{B}^e} N^i \frac{1}{\Delta t} N^j + \nabla N^i \cdot d_{\nabla\phi}\mathbf{q} \cdot \nabla N^j - N^i d_\phi f^\phi N^j dV. \tag{3.10}$$

After the problem is formulated as in (3.10), the Newton-Raphon method can be applied easily by

$$\phi_I \leftarrow \phi_I - \sum_{j=1}^{n_{nd}} K_{IJ}^{\phi,-1} R_J^{\phi}, \quad \text{where} \quad I, J = 1, ..., n_{nd}, \tag{3.11}$$

where n_{nd} is the global node number. The chemical problem is characterized through gating variables, g_{gate}^{I} and g_{gate}^{II}, and ion concentrations introduced in Chapter 2.3. Their advancement in time approximated with the backward Euler method

$$\begin{aligned} \dot{\mathbf{g}}_{gate}^{I} &= [\mathbf{g}_{gate}^{I} - \mathbf{g}_{gate,n}^{I}]/\Delta t, \\ \dot{\mathbf{g}}_{gate}^{II} &= [\mathbf{g}_{gate}^{II} - \mathbf{g}_{gate,n}^{II}]/\Delta t, \quad \dot{\mathbf{c}}_{ion} = [\mathbf{c}_{ion} - \mathbf{c}_{ion,n}]/\Delta t. \end{aligned} \tag{3.12}$$

Then, the update equations for the gating variables can be written in the form,

$$\begin{aligned} \mathbf{g}_{gate}^{I} &= \mathbf{g}_{gate,n}^{I} + \frac{1}{\tau_{gate}^{I}(\phi)}[\mathbf{g}_{gate}^{\infty I} - \mathbf{g}_{gate}^{I}]\Delta t, \\ \mathbf{g}_{gate}^{II} &= \mathbf{g}_{gate,n}^{II} + \frac{1}{\tau_{gate}^{II}(\phi)}[\mathbf{g}_{gate}^{\infty II} - \mathbf{g}_{gate}^{II}]\Delta t. \end{aligned} \tag{3.13}$$

It is important to note that the second group of gating variables depends not only on the current potential value but on the ion concentrations as well. Therefore, the second group of gating variables g_{gate}^{II} are updated iteratively by local Newton iterations. Gating variables control the ionic currents, which change the intracellular ionic concentrations through,

$$\dot{\mathbf{c}}_{ion} = \mathbf{f}_{ion}^{c}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion}). \tag{3.14}$$

Based on the Equation (3.14), the chemical problem can be written as

$$\mathbf{R}_{ion}^{c} = \mathbf{c}_{ion} - \mathbf{c}_{ion,n} - \mathbf{f}_{ion}^{c}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion})\Delta t = 0. \tag{3.15}$$

More specifically, the residual for each ion can be written in the following form,

$$\begin{aligned}
R^c_K =& c_K - c_{K,n} + \frac{C}{VF}[I_{K1} + I_{Kr} + I_{Ks} - 2I_{NaK} + I_{pK} + I_{t0} + I_{stim}]\,\Delta t = 0,\\
R^c_{Na} =& c_{Na} - c_{Na,n} + \frac{C}{VF}[I_{Na} + I_{bNa} + 3I_{NaK} + 3I_{NaCa}]\,\Delta t = 0,\\
R^c_{Ca} =& c_{Ca} - c_{Ca,n} + [\frac{C}{2VF}[I_{CaL} + I_{bCa} + I_{pCa} - 2I_{NaCa}]\\
& - I_{leak} + I_{up} - I_{rel}]\gamma_{Ca}\Delta t = 0,\\
R^c_{Ca} =& c^{sr}_{Ca} - c^{sr}_{Ca,n} + \frac{V}{V^{sr}}[I_{leak} - I_{up} + I_{rel}]\gamma^{sr}_{Ca}\Delta t = 0.
\end{aligned} \tag{3.16}$$

The algorithmic residual is linearized consistently to obtain the matrix $\mathbf{K}^c_{ion\ ion}$. Residual and ion concentration vectors are arranged as $\mathbf{R}^c_{ion} = [R^c_K, R^c_{Na}, R^c_{Ca}, R^{src}_{Ca}]^T$ and $\mathbf{c}_{ion} = [c_K, c_{Na}, c_{Ca}, c^{sr}_{Ca}]^T$, respectively,

$$\mathbf{K}^c_{ion\ ion} = d_{\mathbf{c}_{ion}}\mathbf{R}^c_{ion} = \begin{bmatrix} d_{c_K}R^c_K & d_{c_{Na}}R^c_K & 0 & 0\\ 0 & d_{c_{Na}}R^c_{Na} & d_{c_{Ca}}R^c_{Na} & 0\\ 0 & d_{c_{Na}}R^c_{Ca} & d_{c_{Ca}}R^c_{Ca} & d_{c^{sr}_{Ca}}R^c_{Ca}\\ 0 & 0 & d_{c_{Ca}}R^{src}_{Ca} & d_{c^{sr}_{Ca}}R^{src}_{Ca} \end{bmatrix}. \tag{3.17}$$

The matrix $\mathbf{K}^c_{ion\ ion}$ is used for local iterations at the integration points. For every Newton iteration performed on the problem, the following sets are updated,

$$\begin{aligned}
\mathbf{c}_{ion} &\leftarrow \mathbf{c}_{ion} - [\mathbf{K}^c_{ion\ ion}]^{-1}\mathbf{R}^c_{ion},\\
\mathbf{g}^{II}_{gate} &\leftarrow \mathbf{g}^{II}_{gate} + \mathbf{f}^{gII}_{gate}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion})\Delta t,\\
\mathbf{I}_{crt} &\leftarrow \mathbf{I}_{crt}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion}).
\end{aligned} \tag{3.18}$$

Once the convergence is achieved with the utilization of Newton iterations, the source term $\mathbf{f}^{\phi}(\phi, \mathbf{g}_{gate}, \mathbf{c}_{ion})$ can be calculated from (2.5). The linearized form $d_{\phi}\mathbf{f}^{\phi}(\phi, \mathbf{g}_{gate}$,-$\mathbf{c}_{ion})$ is calculated for the global Newton iteration step in the form as follows,

$$\begin{aligned}
d_{\phi}f\phi =& -[d_{\phi}I_{Na} + d_{\phi}I_{bNa} + d_{\phi}I_{NaK} + d_{\phi}I_{NaCa} + d_{\phi}I_{K1} + d_{\phi}I_{Kr}\\
& + d_{\phi}I_{Ks} + d_{\phi}I_{pK} + d_{\phi}I_{t0} + d_{\phi}I_{CaL} + d_{\phi}I_{bCa} + d_{\phi}I_{pCa}].
\end{aligned} \tag{3.19}$$

The source term and its linearized form are assembled in (3.9) and (3.10), respectively, and then utilized in global Newton iterations. The solution steps of the algorithm are provided in Table 3.1. One significant property of this solution algorithm is that the update formulation for gating variables differs based on whether they depend on the membrane potential. Gating variables depending on the transmembrane potential are updated with local Newton iterations, and the rest of the gating variables are updated with implicit Euler updates for computational efficiency. Local updates are fully implicit in the algorithmic treatment of the problem. Two nested Newton-Raphson algorithms are embedded in the algorithm due to the highly nonlinear nature of the problem. Notably, the number of equations, the linearization processes, and the inversion of linearization matrices are computationally demanding steps.

Table 3.1: Algorithmic approach for the electrochemical problem of the cardiac tissue based on finite element discretization in space and finite difference discretization in time. Two nested Newton-Raphson iterations are employed to converge to a solution. The membrane potential is the global degree of freedom ϕ and gating variables g^{II}_{gate}, g^{II}_{gate}, and ion concentrations c_{ion} are solved locally at the integration point [1].

initialize degree of freedoms at nodes ϕ
initialize gating variables $\mathbf{g}^{I}_{gate}$, $\mathbf{g}^{II}_{gate}$
initialize ion concentrations $\mathbf{c}_{ion}$
global Newton iteration
loop over all elements
loop over all integration points
update first set of gating variables $\mathbf{g}^{I}_{gate} \leftarrow \mathbf{g}^{I}_{gate} + \mathbf{f}^{gI}_{gate}\Delta t$
initialize second set of gating variables $\mathbf{g}^{II}_{gate} \leftarrow \mathbf{g}^{II}_{gate} + \mathbf{f}^{gII}_{gate}\Delta t$
initialize ionic currents $\mathbf{I}_{crt} \leftarrow \mathbf{I}_{crt}(\phi, \mathbf{g}^{I}_{gate}, \mathbf{g}^{II}_{gate}, \mathbf{c}_{ion})$
local Newton iteration
calculate ion concentration residuals $\mathbf{R}^{c}_{ion}$ and local iteration matrix $[\mathbf{K}^{c}_{ion\ ion}] = d_{\mathbf{c}_{ion}}\mathbf{R}^{c}_{ion}$
update ion concentrations $\mathbf{c}_{ion} \leftarrow \mathbf{c}_{ion} - [\mathbf{K}^{c}_{ion\ ion}]^{-1}\mathbf{R}^{c}_{ion}$
update second set of gating variables $\mathbf{g}^{II}_{gate} \leftarrow \mathbf{g}^{II}_{gate} + \mathbf{f}^{gII}_{gate}\Delta t$
update ionic currents $\mathbf{I}_{crt} \leftarrow \mathbf{I}_{crt}(\phi, \mathbf{g}^{I}_{gate}, \mathbf{g}^{II}_{gate}, \mathbf{c}_{ion})$
calculate source term $f^{\phi}(\mathbf{I}_{crt})$ and its linearization $d_{\phi}\mathbf{f}^{\phi}$
calculate element residuals $\mathbf{R}^{\phi^e}_{I}$ and element matrices $\mathbf{K}^{\phi^e}_{IJ} = d_{\phi^e_j}\mathbf{R}^{\phi^e}_{I}$
calculate global residual $\mathbf{R}^{\phi}_{I}$ and global iteration matrix $\mathbf{K}^{\phi}_{IJ} = d_{\phi_J}\mathbf{R}^{\phi}_{I}$
update membrane potential $\phi_J \leftarrow \phi_J - \mathbf{K}^{\phi-1}_{IJ}\mathbf{R}^{\phi}_{I}$

CHAPTER 4

DEEP LEARNING

This chapter introduces deep learning, some important layer types, and the backpropagation algorithm. Last, the proposed model in the framework, chosen hyperparameters, and the loss function are presented.

4.1 Introduction to Deep Learning

Deep learning (DL) is a subfield of machine learning that emphasizes learning increasingly meaningful representations from data using numerous interconnected layers. The number of layers contributing to a model is called the depth of the model [73]. The most prominent property of DL is that it can extract highly complex features, unlike most machine learning algorithms. In DL, learned features are progressively refined through iterative learning processes, resulting in higher-level representations of the data. These higher representations are learned by neural networks most of the time. The term neural network is inspired by neurobiology by associating the layers in neural networks with the nerve cells. Generally, neural networks consist of layers stacked on top of each other. Layers of the network collaborate to transform an input progressively into abstract representations.

In recent years, the popularity and applicability of DL have become widespread among all scientific fields. Multiple factors have caused the sudden increase in the usage of DL algorithms. Modern hardware has orders of magnitude more computational power than their predecessors. Fast and massively parallel chips have been developed by companies, and their accessibility to researchers has made a considerable contribution to the DL field. An increase in the data acquisition technologies

and storage capacities resulted in massive available data sets that DL models need. Another important step is the discovery of algorithmic improvements. Many DL concepts still used today have been known for a long time. For example, some fundamental concepts in DL, such as convolutional neural networks and Long Short-Term Memory algorithm, were discovered before the 2000s. However, the critical issue of DL, the vanishing gradient problem, has stalled the progress in the field. The proposed improvements in the early 2010s, such as better activation functions, optimization schemes, and weight initialization schemes, have significantly contributed to this problem's solution. These improvements have resulted in deeper and more capable neural networks. Also, financial investments in the field have boosted the progress. In 2011, the total venture capital investment in AI was around $19 million; by 2014, the investment had risen to $394 million. Finally, DL tools have become accessible to large groups of people, the need for specialized equipment and significant expertise has decreased, and the number of people studying the subject has drastically increased [73].

DL has been observed to exhibit remarkable success across a multitude of diverse problems. The most prominent breakthroughs of DL are near-human-level image classification, near-human-level speech recognition, near-human-level handwriting transcription, improved machine translation, near-human-level autonomous driving, and the ability to answer natural-language questions. For the computational modeling and its application in the heart, we refer to [74] and references therein.

DL can be grouped into three categories based on the applied learning technique: supervised, unsupervised, and reinforcement learning. Supervised learning is defined by its use of labeled data and training the model according to the labels to solve a classification or a regression problem. In unsupervised learning, training data is unlabelled, and DL algorithms focus on discovering data patterns and creating data groupings. Finally, in reinforcement learning, the algorithm is not explicitly programmed to which actions it should take; instead, the model discovers the appropriate actions by maximizing the reward based on the outcomes [75].

The DL model employed in this work is a supervised learning model. The model predicts the action potential in the cardiac tissue at the cellular level, and therefore, it

can be classified as a regression problem. It should be noted that a DL model is not comprised solely of layers that construct it. The choice of the hyperparameters, the loss function, and the order of layers play a significant role in the performance of the model. In the following sections, this work delves deeper into the DL environment. The structure of the proposed model's architecture is explained, and the proposed model is introduced to the reader.

4.2 Deep Learning Model

In this section, fundamental aspects of the deep learning architecture will be introduced, and the employed model will be demonstrated. Neurons are fundamental computational units of DL layers. Number of neurons in each layer can go up to thousands. Every neuron performs a specific action to its inputs. First, elementwise multiplication is performed between inputs and neuron weights, and weighted inputs are summed. In general, the neurons have a bias term, which is also included in the summation. These bias terms allow shifting the sum in a specific direction and simplifies the learning process. The resulting value constitutes the input value for the neuron's activation function. As the final step, the output of the activation function is forwarded deeper inside the NN framework.

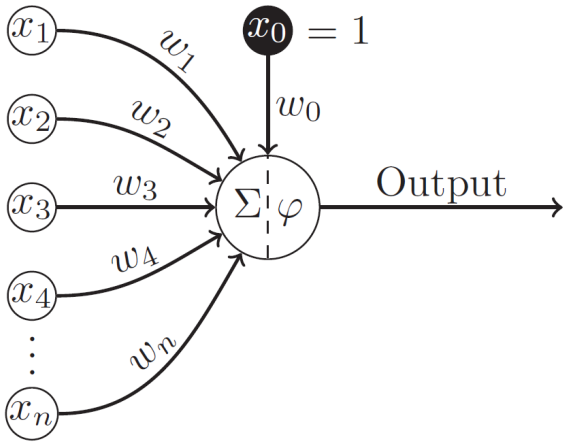

Figure 4.1: Structure of a neuron with the bias term [3].

In Figure 4.1, the general structure of a neuron is represented. While the x_i with $i =$

$1, ..., n$ terms represent inputs, w_i with $i = 1, ..., n$ terms represent the neuron's weights. Also, the bias term is included as w_0. Neuron has two functionalities, which are summation and activation. The operations within a neuron can be formulated as,

$$output = \sigma(\sum_i w_i x_i + w_0), \tag{4.1}$$

where σ is the activation function of a neuron. Activation functions are crucial in a DL framework and can be a significant factor in the learning speed and accuracy of the model.

4.2.1 Activation Functions

The principal role of the activation function is to introduce nonlinearities to the outputs of the individual neurons to capture complex patterns in the data. Activation functions impose nonlinearities and some restrictions on the outcome of the neuron. These restrictions can include limiting neuron outcomes within specific ranges or deciding if a neuron is active.

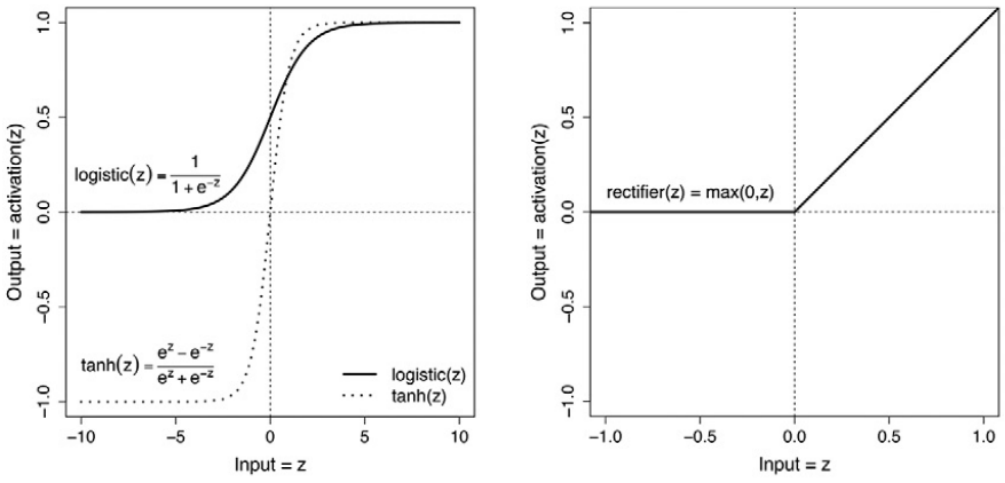

Figure 4.2: Some common activation functions [3].

Figure 4.2 represents some common activation types utilized in DL. The logistic activation function, also called the sigmoid function, transforms the output in a range between $[0, 1]$, while the outcome of the tanh function is between $[-1, 1]$. Finally, the

rectifier activation function, also called rectified linear unit, is one of the most popular choices for activation functions, with a minimum limit of 0 for outcome and no upper limit. The main reason for the rectifier function's popularity is that the derivative of this activation function does not saturate. In sigmoid and tanh functions, the function's derivative becomes close to 0 for the values larger than 3 and smaller than -3. However, the derivative is a constant value for positive inputs in a rectified linear function. This fact becomes a pivotal point in training the NNs. Once the backpropagation algorithm is explained, the derivative's importance will be more evident in the following sections.

4.2.2 Fully Connected Layer

A fully connected layer, also called a dense layer, is a fundamental DL layer type. Each neuron from the previous layer has direct connections to the neurons of the ensuing fully connected layer.

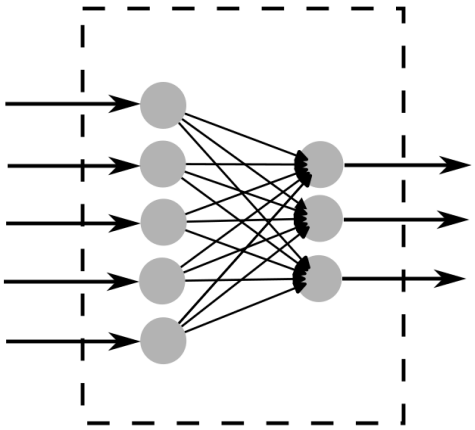

Figure 4.3: A representation of fully connected layer.

All possible connections between the fully connected layer and its inputs layer are present, as shown in Figure 4.3. This connection pattern means every input affects the output of every neuron of the fully connected layer.

4.2.3 Convolution Layer

Convolutional layers are specifically designed layers for processing data that have grid-like topology. By employing kernels, the convolutional layer extracts features from the input layer. The most important characteristic difference between convolutional layers and fully connected layers is sparsity. In the convolutional layers, input layers do not connect with every neuron of the convolutional layer. The convolution operation is shown to be extremely successful in the DL world. Convolutional layers leverage some distinct ideas to improve learning. They have sparse interactions between input neurons, and the same parameters are utilized for more than one function [76].

One of the convolution's most characteristic features is its success in extracting features from spatial relations. The filters' number and shape are hyperparameters that are decided while designing a convolutional layer. Convolutional layers are generally used together with other types of layers, such as pooling and batch normalization.

4.2.4 Recurrent Layer

A recurrent layer is specifically designed to process and model sequential or time series data effectively. The distinguishing characteristic of recurrent layers is that they can capture temporal dependencies. In other traditional layers, the input and output of the model are independent from the previous inputs. In recurrent layers, however, a dependence is achieved by incorporating recurring connections to the layer. Due to its recurrent connection, these layers share the parameters across the input sequence.

Figure 4.4 demonstrates the structure of a recurrent layer. Hidden state variables inside the recurrent layer are repeatedly fed back to the layer through the sequence. In general, recurrent layers are unrolled in time to demonstrate the structure of the layer. An adaptation of a backpropagation algorithm, the backpropagation through time (BPTT), is used to train recurrent layers.

The process inside a recurrent layer is shown in Figure 4.5. h_{t-1} is the hidden state of the previous layer, and x_t is the model's input at a given time. Merging of the paths

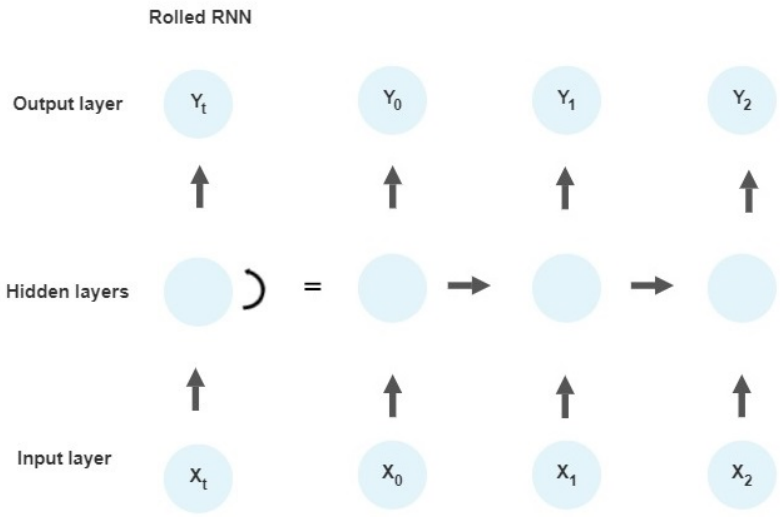

Figure 4.4: Rolled and Unrolled demonstration of an RNN.

means a concatenation between vectors, and $tanh$ is the layer's activation function. The terms o_t is the layer's output, and h_t is the new hidden layer that will be forwarded to the next step. One weakness of the recurrent layers is that they cannot learn long sequences and have problems with the gradient propagation during the training phase. Two other layers with recurrent connections are proposed to address this problem: Gated Recurrent Units (GRU) [77] and Long Short-Term Memory (LSTM) [78].

LSTMs have multiple paths and gates within themselves. They are specifically designed to learn long sequences, with fewer problems with gradient backpropagation. The cell state c in an LSTM layer is subjected to fewer algebraic operations to ease the gradient backpropagation. The cell state represents the long-term memory, while the hidden state h represents the short-term memory. In Figure 4.6 c_{t-1} and h_{t-1} are, respectively, the cell state and hidden state variables from the previous step. The operations $+$, $1-$, and $\times$ are elementwise. Additionally, the sigmoid activation function σ takes place within the layer. The rest of the variables are current cell state c_t, current hidden state h_t, input x_t, and internal states f_t i_t $\tilde{c}$ and o_t. LSTMs are computationally more expensive than recurrent layers. However, they can generally model longer and more complex sequences. An LSTM layer is employed by the DL model proposed in this study. On the other hand, GRUs are slightly modified versions of LSTMs.

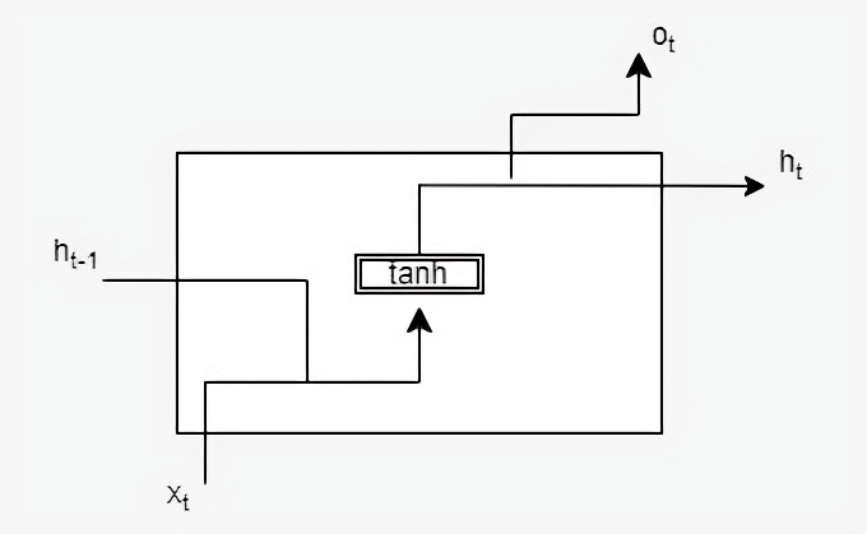

Figure 4.5: Structure of a recurrent layer.

A GRU layer has fewer gates and variables compared to LSTM. In Figure 4.7, the structure of the GRU layer is demonstrated. The hidden state of the GRU operates as a combination of short and long-term memories. The performance of LSTMs and GRUs depends on the problem, while both perform better than the simplistic recurrent layer in general.

4.2.5 Backpropagation Algorithm

Backpropagation, short for "backward propagation of errors," is an essential algorithm for training DL models. Backpropagation performs an automatic differentiation technique to complex nested functions. The popularity of the backpropagation algorithm surged after the discovery of its applicability to NNs. Given an NN and a loss function, the backpropagation algorithm calculates the gradient of the error function with respect to the weights of the NN. Gradients are calculated from the last layers of NN and propagated toward the initial layers through the chain rule of differentiation.

Training a NN with backpropagation follows several steps. Firstly, an error calculation function is determined. The choice of this function is based on the NN's purpose and the data's characteristics. For example, a mean squared error loss function computes the loss as the square of the difference between the output of the NN and the

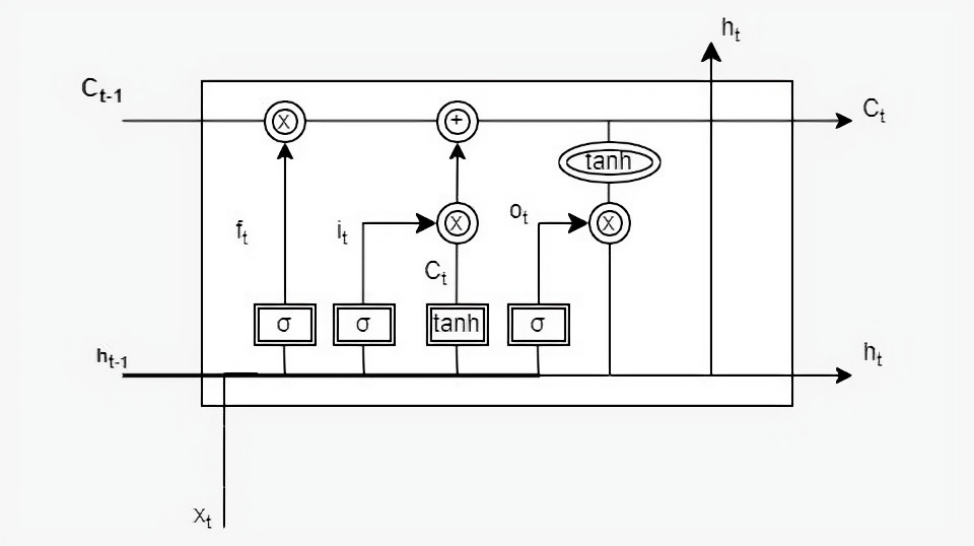

Figure 4.6: Structure of an LSTM layer.

actual result (label). In the forward pass, input data is passed from layer to layer, and an outcome is obtained. A loss value is calculated with the loss functions. In the backward pass, the backpropagation algorithm calculates the gradients of loss with respect to the model parameters. Parameters of the model are updated in the opposite direction of the gradient to minimize the error. In the update step, a gradient descent-based algorithm is employed generally. By repeating these steps, the model is trained until the error is below a determined tolerance or until the time and computation restrictions are exceeded.

4.2.6 Proposed Model

The model is created with the PyTorch framework. Pytorch is a machine learning framework based on the Torch library [79]. This framework aims to accelerate the development of machine learning models. PyTorch offers DL-specific layers, loss functions, and optimization algorithms. The version of the PyTorch framework utilized in this study is 1.12.1 with CUDA support. The parameters not explicitly mentioned in this study are the default values from the implemented PyTorch framework. CUDA (Compute Unified Device Architecture) is a parallel computing platform and application programming interface created by NVIDIA. It allows developers to harness the computational power of NVIDIA GPUs (Graphics Processing Units) for general-

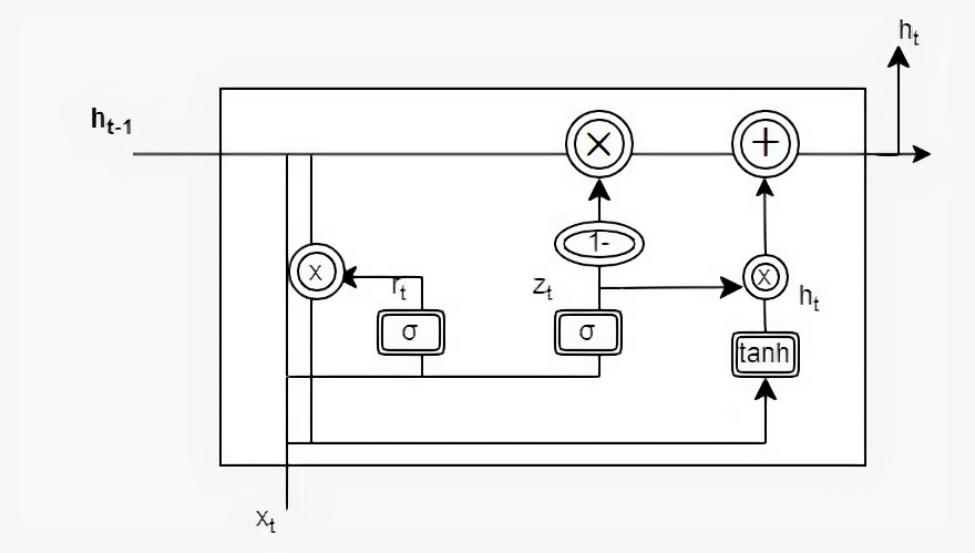

Figure 4.7: Structure of a GRU layer.

purpose computing tasks other than graphical rendering.

The proposed model in this study is demonstrated in Figure 4.8. The input of the model has the shape of $[n_{feature}, n_{sequence\ length}, n_{depth} = 1]$. The term $n_{feature}$ is the number of input features at each sequence step, while $n_{sequence\ length}$ represents the length of the respective part of the electrophysiological signal. The depth of the raw input signal n_{depth} is one, as only the features and sequence length are determined and needed. The architecture of the model can be divided into three parts. The first part consists of convolutional layers, activation functions, and dropout layers. This part is followed by an LSTM layer to process sequential information of the signal. Finally, three fully connected layers process the signal and output a single prediction value.

The first part with the convolutional layers functions as an autoencoder, where a combination of convolutional and pooling layers encodes the input signal. The purpose of convolutional layers is to extract intricate features and patterns from raw input data automatically. A total of five convolution layers are proposed in the model architecture. Four filters are applied to the input signal at the first convolution layer. In the following layers, twice the number of filters applied in the preceding layer are applied. At the end of the convolution part, the depth of the forwarded input becomes $n_{depth=64}$. The filters' width is aligned with the input's feature dimension, and the filters' height is aligned with the sequence dimension. Convolution filters have a height

of 3, which means each filter seeks patterns for sequence lengths of 3. The filters' width is one except for the fourth layer, and the filters' depth is equal to the depth of the input at the current position. Filters with a width of one extract feature from a single input, while the fourth layer combines all inputs and creates features with information from all inputs. Each filter has a stride of one in both directions, which means filters move without any jumps. The size of the output of a convolution layer can be calculated for both feature and sequence dimensions as follows,

$$n_{out} = [\frac{n_{in} + 2p - k}{s}] + 1, \tag{4.2}$$

where n_{out} is the output features, n_{in} is the input features, s is the stride size for the kernel in the given direction, and p is the padding for input data. This model's data is padded with zeros in the sequence dimensions to keep $n_{in} = n_{out}$. Padding with zeros is a generally accepted approach in the DL community. Zero padding has the benefit of considering the last features of the input multiple times, as they carry significant information about the state of the signal most of the time. The output of the convolutional layers is forwarded to pooling layers except for the last convolutional layer. The purpose of the pooling layers is to decrease the sequence length by keeping the outputs of the most active neurons. Pooling filters have a stride of 2 in the sequence dimension, meaning they halve the sequence dimension length and present a more compact output. The outputs are forwarded to a leaky-ReLU activation function to introduce nonlinearities after every convolutional layer. The leaky-ReLU activation function is defined as,

$$LeakyRELU(x) = \begin{cases} x, & \text{if } x \geq 0, \\ 0.1x, & \text{otherwise.} \end{cases} \tag{4.3}$$

The second part of the model has an LSTM layer. The purpose of this layer is to process the encoded form of the signal as a time series with recurrent connections. The most critical parameter in the LSTM layer is the number of units n_{hidden} in the hidden state. 400 hidden units are present in the proposed architecture for carrying the time series information in an accurate power with necessary modeling capabilities.

The final part of the NN consists of fully connected layers. Three fully connected layers are stacked on each other to increase the prediction capabilities of the NN. The number of neurons in these layers is kept relatively small compared to modern NN models for keeping the computational load at an optimum level. While the first fully connected layer has 400 neurons, the following layers have 200 and 100 neurons. The last fully connected layer generates a single output. A ReLU activation layer is present after fully connected layers except for the last layer. Since the model aims to predict a value from an unbounded range, an activation function is absent after the last fully connected layer.

The output of the model and label of the data are fed into the chosen loss function. The chosen loss function in this model is the minimum squared error (MSE) loss function,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}[y_i - \tilde{y_i}]^2, \tag{4.4}$$

where N is the number of predicted outputs, and $\tilde{y_i}$ is the label of the input. The gradient of the loss function with respect to model parameters is backpropagated, and then the model parameters are updated with the Adam optimization algorithm [80]. Adam stands for 'Adaptive Moment Estimation,' and it is an extended version of a stochastic gradient descent optimizer. This optimizer dynamically adjusts learning rates based on gradient moments, aiming to accelerate the decrease of the loss value and improve the training process of the DL model.

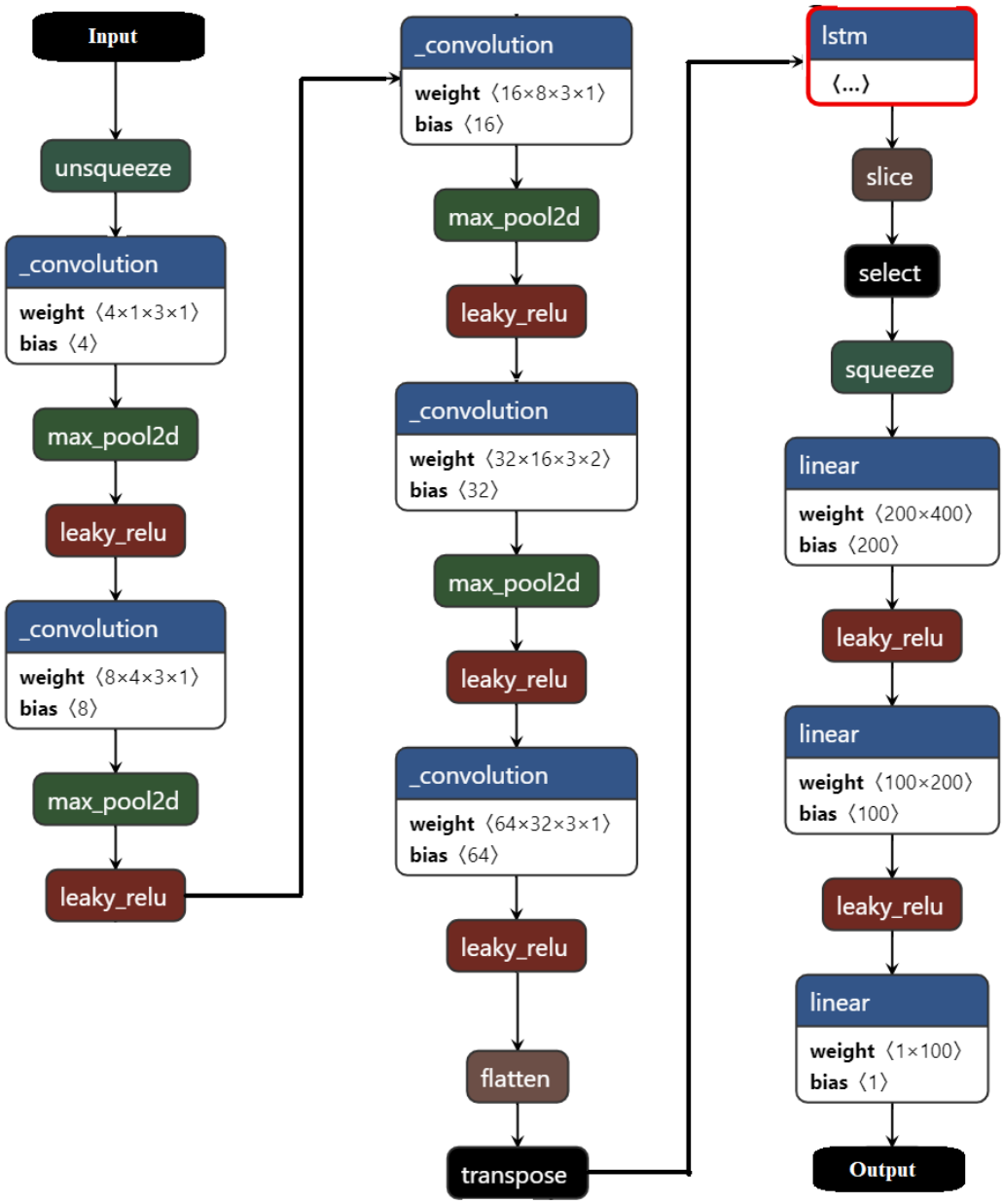

Figure 4.8: Architecture of the proposed model.

CHAPTER 5

ORDINARY DIFFERENTIAL EQUATIONS AND DEEP LEARNING

In this chapter, training the DL model based on the ten Tusscher-Panfilov model will be explained in detail. Training data are generated from the solutions of the ten Tusscher-Panfilov model as an ODE system. The architecture of the DL model, preprocessing, and training stages are discussed in Chapter 2.

5.1 Data Generation

The general form of the local electrophysiological problem can be described as follows,

$$\dot{\phi} + f^{\phi} + I_{ext} = 0 \tag{5.1}$$

in the ODE setting. f^{ϕ} is the source term resulting from the electrical response at the cellular level, $\dot{\phi}$ is the rate of change of the potential, and I_{ext} is the external stimulus. The ten Tusscher-Panfilov model parameters and initial state values are given in Table 2.2. The model can provide accurate response up to time steps as large as 0.2 ms [1]. In this study, time step Δt is chosen as 0.08 ms for data generation unless stated otherwise. The initial potential value of the model ϕ_0 is defined as -86.2 mV, approximately equal to the resting potential for a cardiac myocyte in the ten Tusscher-Panfilov model. The algorithm created for the ten Tusscher-Panfilov model returns the value of ϕ when provided with ϕ_n and I_{ext} values. This process can be demonstrated as $Tusscher(\phi_n, I_{ext}) \rightarrow \phi$. The external stimulus I_{ext} is the stimulation provided to the ODE externally between $[t_n, t_{n+1}]$. The pair $[\phi_n, I_{ext}]$ is

saved to a data set for every time step. The unit for all stimuli in this section is mV/ms.

Training a DL model to capture the possible complex phenomena occurring in the ten Tusscher-Panfilov model requires an extensively large data set. Two important model parameters are randomly selected from uniform distributions to reflect these complex behaviors. These are the magnitude of the given stimulus I_{ext} and the time between the consecutive stimuli that can generate the action potential. The given stimuli to the model are divided into two groups.

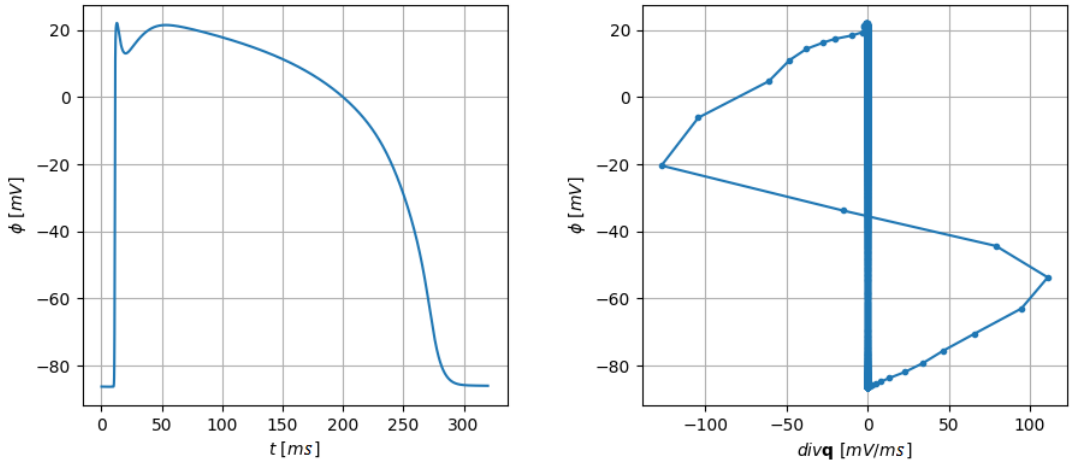

Figure 5.1: Action Potential ϕ cycle of a cell from a PDE solution with respect to time t on the left, and Potential Value ϕ with respect to divergence term div$\mathbf{q}$ in the PDE (right).

The first group of stimuli is directly associated with the excitation of the cell. Figure 5.1 presents the potential value of a cell with respect to time and divergence term. The data for the figure are obtained from a PDE solution in the 2-D domain with the utilization of ten Tusscher-Panfilov model in the source term. The data point is chosen at a central location of the domain as an exemplary point that exhibits standard behavior. During the depolarization phase, the divergence term exhibits significant variations; in the figure, the points in which div$\mathbf{q}$ deviates significantly from zero correspond to the depolarization stage. These large variations in the depolarization stage are considered in deciding the intervals of the I_{ext} terms. The initial I_{stim1} term is bounded between $[25, 120]$ mV/ms values, and it is applied for the duration t_{stim1} within the range of $[0.5, 1.25]$ ms. The second part of the external stimuli I_{stim2} is

within the range of $[-50, 50]$ mV/ms, and the duration t_{stim2} of it is within $[1.25, 2.5]$ ms. All the values are selected randomly from uniform distributions with the given bounds. The process can be summarized as follows; first I_{stim1}, I_{stim2}, t_{stim1} and t_{stim2} values are selected. At the first stage the stimulus I_{stim1} is applied until t_{stim1}. In the second stage, the stimulus I_{stim2} is applied until t_{stim2} and the excitation stage is completed. A significant property for the stimuli values I_{stim1} and I_{stim2} and, respectively, their intervals $[0, t_{stim1}]$ and $[t_{stim1}, t_stim2]$ is that they do not guarantee the initiation of depolarization, since selected stimulus values may not provide the necessary stimulation to initiate the action potential cycle with the given magnitude and duration intervals.

$$\begin{aligned} &I_{stim1} \sim U(25, 120) \quad \text{with} \quad t_{stim1} \sim U(0.5, 1.25), \\ &I_{stim2} \sim U(-50, 50) \quad \text{with} \quad t_{stim2} \sim U(1.25, 2.5), \\ &I_{stim} = I_{stim1} \quad \text{while} \quad t \in [0, t_{stim1}], \\ &I_{stim} = I_{stim2} \quad \text{while} \quad t \in [t_{stim1}, t_{stim2}]. \end{aligned} \tag{5.2}$$

The equation (5.2) shows the formulation for the exciting stimuli group, where $U(\cdot, \cdot)$ is the notation for the uniform distribution. The selected values' intervals have been kept as wide as possible to create a greater variety of conditions in the training of the model. The aim of training the model in this manner is to enhance its capacity for extrapolation. The time between the start of the exciting stimulus and the starting time of the following exciting stimulus is chosen from the range $[250, 320]$ ms. This interval is chosen since high-frequency excitations create unique and challenging phenomena. The model can show complex reentry behavior if the excitation occurs shortly after the end of the effective refractory period. Moreover, the action potential duration (APD), which is the time needed for a cell to undergo depolarization and repolarization phases, changes with the excitation frequency. As the frequency of excitation stimuli increases, the action potential duration of the cell decreases. Another important reason for choosing this range of values is that since all possible values for external stimuli will not start a depolarization, the interval between the consecutive depolarization times can be larger than 320 ms. Therefore, the data set will implicitly include two consecutive waves with large time intervals between them.

The second group of stimuli plays the role of noise. The numerical method and precision of computation can cause noise. Also, some minor stimuli can be applied to the model purposefully. The noise stimuli do not have a time interval; instead, they can be introduced in any step unless an exciting stimulus is being applied. In the problem, the noise stimuli are divided into three groups. The magnitude of the first stimulus group ranges between $[-10, 10]$ mV/ms and occurs with a probability of 0.5% at a time step. The second group has a magnitude range of $[-0.1, 0.1]$ mV/ms with a probability of occurrence of 24.5%. The magnitude of the final group is between $[-0.001, 0.001]$ mV/ms with a probability of occurrence of 75%. Noise stimulus can be shown in the following way,

$$\begin{aligned} I_{noise1} &\in [-10, 10] && \text{with} && P(I_{noise1}) = 0.005, \\ I_{noise2} &\in [-0.01, 0.01] && \text{with} && P(I_{noise2}) = 0.245, \\ I_{noise3} &\in [-0.001, 0.001] && \text{with} && P(I_{noise3}) = 0.750, \end{aligned} \tag{5.3}$$

where $P()$ is the probability of occurrence of a noise stimulus. It should be noted that noise will be present at every time step since a minor noise is likely to occur in a numerical solution of a PDE. The magnitude intervals for I_{noise2} and I_{noise3} terms are determined based on Figure 5.1. I_{noise1} values represent larger noises that can occur due to several reasons. For example, they can result from the geometry or be provided in the stimulation. The range interval for I_{noise1} is significantly larger than I_{noise2} and I_{noise3}, and they do not disturb the stimulation extensively. All noise terms are incorporated into data to enhance the robustness and reliability of the DL model.

In Figure 5.2, the initial part of the generated data set with the application of noise and external stimuli is demonstrated. In the depolarization phase, the membrane potential gradient exhibits a rapid ascent towards significantly large magnitudes in a very short temporal interval. This characteristic behavior of a cell's action potential imposes a significant challenge in modeling and necessitates the usage of short time steps for conventional numerical methods due to stability problems. Accurate prediction of potentials in the depolarization phase by the NN model is a critical goal in the model development since depolarization has major effects on the rest of the action potential phases. However, the depolarization phase constitutes a tiny fraction of the action

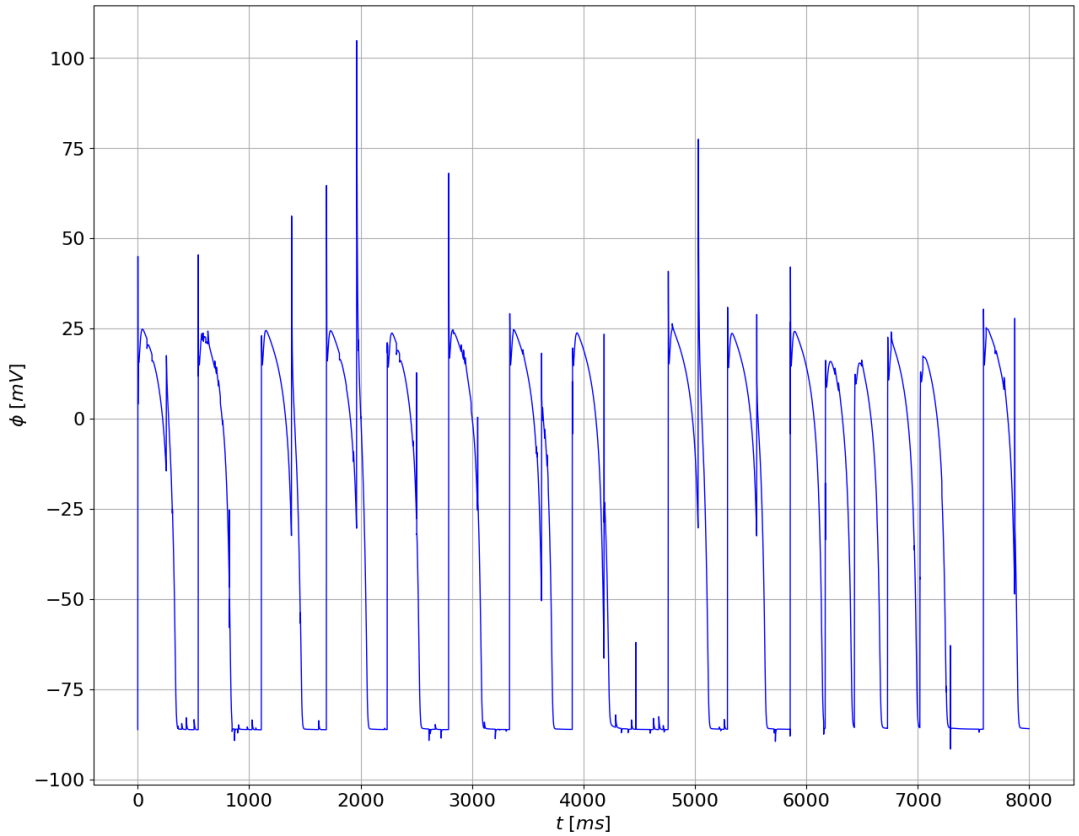

Figure 5.2: Potential values with respect to time from a frame of the data set where only the first 8000 ms are considered.

potential cycle. This results in an unbalanced data set where the depolarization phase is underrepresented in the data set.

The training data set is enriched by adding data that only involves potentials from the depolarization and initial repolarization phases. These minor data sets added to the training data are called attention sets. These attention sets are obtained by solving the problem for short periods that are just enough to capture the third wave's depolarization and initial repolarization with adequate history data to include in the sequence. Figure 5.3 demonstrates the area where data points will be included in the training set in the red box. Previous points from the red box are present in the sequence; however, only those points inside the red box are utilized in training. Adding these attention sets improved the overall performance for prediction and robustness of the model.

One major advantage of the DL model is that it can predict results for larger time

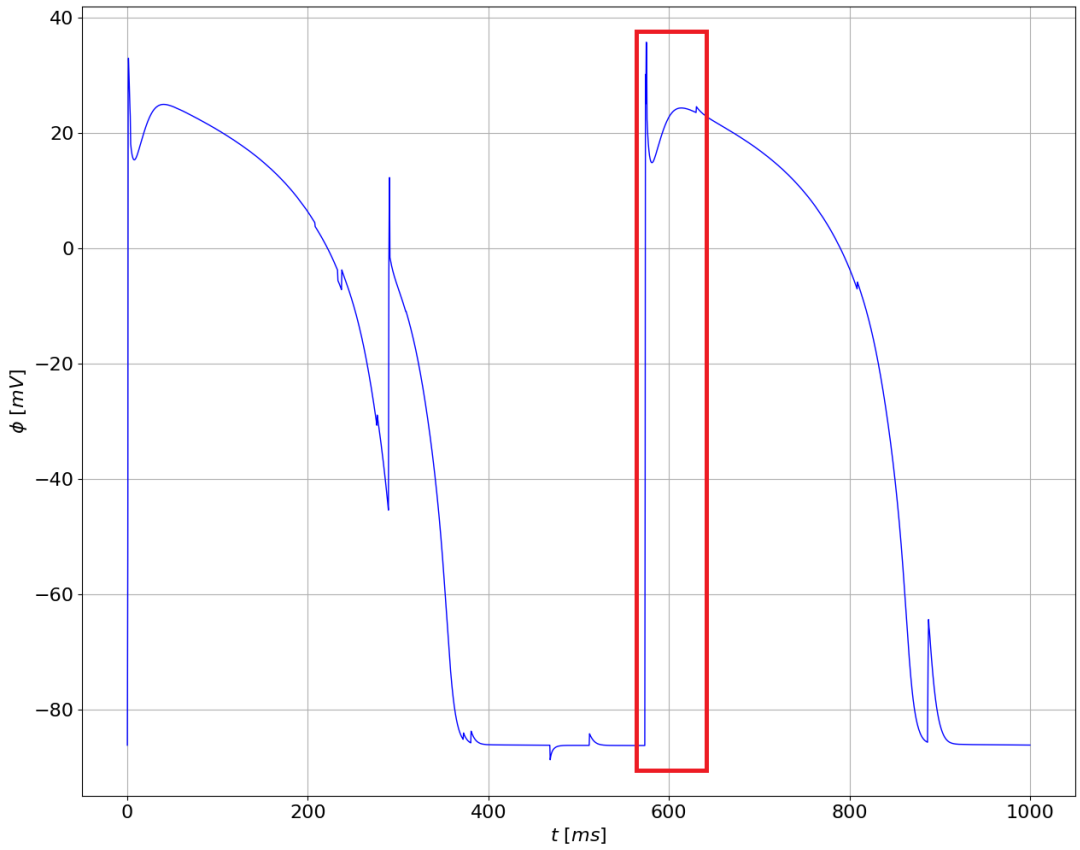

Figure 5.3: Potential values with respect to time from an attention data set, the depolarization, and initial repolarization data are demonstrated inside the red box.

steps in a data-driven way and is less likely to suffer from instability problems like the numerical methods. Therefore, it is possible to utilize the DL to work with larger time steps than the ten Tusscher-Panfilov model is capable of. In this study, unless stated otherwise, the model is trained to predict ϕ value with time steps of 0.40 ms. Some adjustments are applied to the data sets to achieve predictions with larger time steps. Since the ten Tusscher-Panfilov model is solved with time steps of 0.08 ms, external stimuli are kept constant for the model's chosen time step, 0.40 ms, to achieve consistency over the data sets. The process of formatting the data set in the desired manner can be demonstrated in the following form,

$$\begin{bmatrix} I_{ext}^{0} & \phi^{0} \\ I_{ext}^{0} & \phi^{1} \\ I_{ext}^{0} & \phi^{2} \\ I_{ext}^{0} & \phi^{3} \\ I_{ext}^{0} & \phi^{4} \\ I_{ext}^{1} & \phi^{5} \\ I_{ext}^{1} & \phi^{6} \\ \vdots & \vdots \\ I_{ext}^{n} & \phi_{ext}^{m} \end{bmatrix} \xrightarrow[Adjusted\ Data]{} \begin{bmatrix} I_{ext}^{0} & \phi^{0} \\ I_{ext}^{1} & \phi^{5} \\ I_{ext}^{2} & \phi^{10} \\ I_{ext}^{3} & \phi^{15} \\ I_{ext}^{4} & \phi^{20} \\ I_{ext}^{5} & \phi^{25} \\ I_{ext}^{6} & \phi^{30} \\ \vdots & \vdots \\ I_{ext}^{n} & \phi_{ext}^{\lfloor m/5 \rfloor} \end{bmatrix}. \tag{5.4}$$

Therefore, the generated data becomes consistent with external stimuli and can be represented by large time steps.

The training data is generally partitioned into three distinct groups when training DL models. The first group is the training data, where the model parameters are adjusted according to them with the backpropagation algorithm. The purpose of the validation set is to adjust the model's hyperparameters, such as the optimization algorithm's learning rate, number of epochs, number of layers, and number of neurons in each layer. Finally, the model's final performance is measured by how well it works with the test set since it has never encountered the data from the test set. Overfitting is a common problem in DL since DL models can possess significant potential for learning highly complex data. Overfitting occurs when the DL model memorizes the training data and is not able to generalize to other data of the same problem. In this

problem, overfitting is not a major issue since the data is generated from a structured ODE system. A DL model's generalization capability is assessed by its performance on the test set. Three different data sets are created to train the proposed model in this study. The ODE system is solved for 72000 ms for the training data set, and for validation and test sets, the system is solved for 10000 ms. The dataset obtained from a solution of 72000 ms is sufficient for training since it exhibits all the necessary behavior that needs to be learned by the model, while datasets obtained from 10000 ms solutions are sufficient for hyperparameter adjustment and model performance analysis. Additionally, the model is solved for 1000 ms for each generated attention data set.

Prior to training, some preprocessing is applied to raw data. The scale of the range of the input variables differs significantly in the generated data. As a general practice in DL, each input is standardized to have values similar in the scale. The reason for this standardization is to prevent an input from having an unbalanced effect on the model's output and to ease the learning process by scaling the gradients. The standardization is performed for every single feature value,

$$\overline{x_i} = \frac{x_i - \mu}{\sigma}, \tag{5.5}$$

with μ is the mean σ is the standard deviation for the feature, and x_i is the unstandardized value of the feature. The mean and standard deviation values are calculated from the training set data only without attention sets and used for standardizing the data of attention, validation, and test sets. The last preprocessing step is creating input and label pairs from raw data. This process can be demonstrated as follows,

$$raw\ data = \begin{bmatrix} I_{ext}^0 & \phi^0 \\ I_{ext}^1 & \phi^1 \\ I_{ext}^2 & \phi^2 \\ I_{ext}^3 & \phi^3 \\ I_{ext}^4 & \phi^4 \\ \vdots & \vdots \\ I_{ext}^s & \phi^s \end{bmatrix}_{(s+1)\times i} \rightarrow input = \begin{bmatrix} I_{ext}^0 & \phi^0 \\ I_{ext}^1 & \phi^1 \\ I_{ext}^2 & \phi^2 \\ I_{ext}^3 & \phi^3 \\ I_{ext}^4 & \phi^4 \\ \vdots & \vdots \\ I_{ext}^{s-1} & \phi^{s-1} \end{bmatrix}_{s\times i}, \quad label = \begin{bmatrix} \phi^s \end{bmatrix}_{1\times 1}, \tag{5.6}$$

where s is the data sequence length, and i is the number of input features. The indices in raw data in (5.4) are reinitialized in (5.6). Before training, input and label pairs are randomly shuffled to prevent the model from learning from consecutive data. The model has been trained to predict the subsequent membrane potential value at one time step ahead when provided with past stimuli and potential values in its history as inputs.

5.2 Training of the Model

The proposed model is trained for 300 epochs, and the model's parameters are updated with the Adam optimizer. The initial learning rate of the optimizer is 0.0001; however, it is updated at epochs 150, 200, 250, and 275. These milestones are determined by training the model and observing the change in the loss values. Once the epoch number from one of the stated milestones is reached, the learning rate is multiplied by $\gamma = 0.2$. The reason for decreasing the learning rate is to reduce the chance of unexpected behavior of the model in the long term and to increase the overall performance.

In the training, the minimum squared error loss function calculates the loss value for each model output. Figure 5.4 demonstrates the change of loss values over epochs. The effects of the decrease in the learning rate are visible in the figure, with sharp declines in the loss values. Learning rates are multiplied by $\gamma = 0.2$ at the epochs where the decrease rate of the loss starts to stagnate. During the backpropagation,

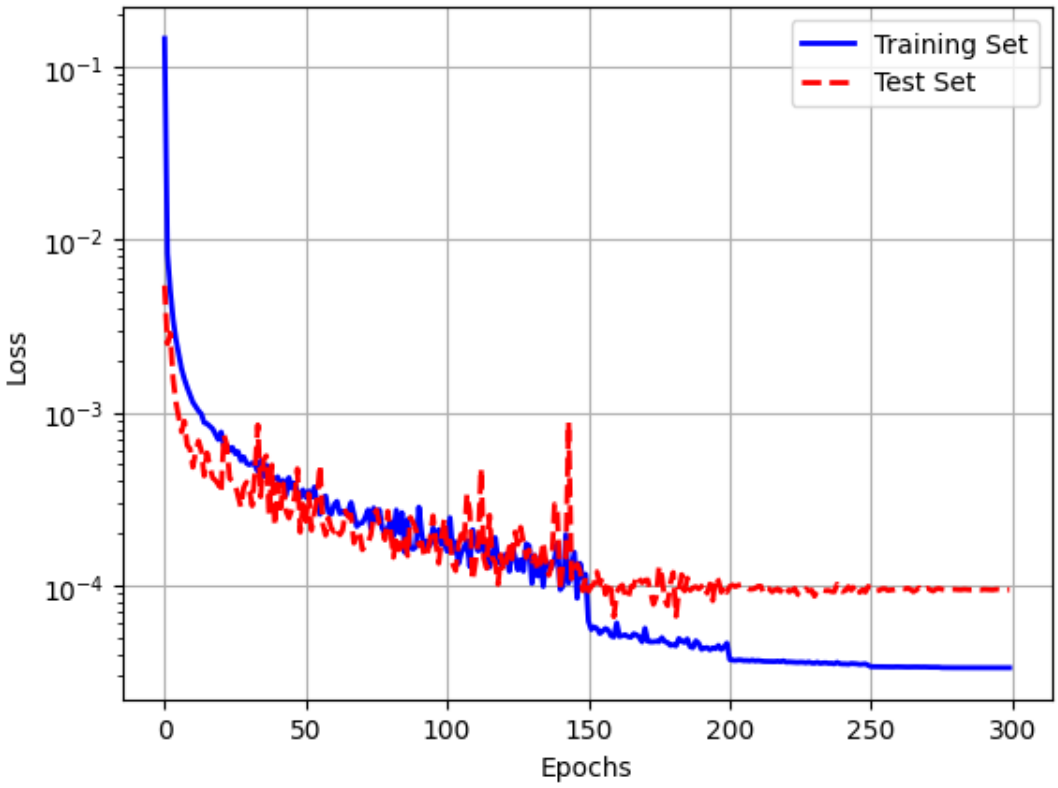

Figure 5.4: Change of loss values with epoch number for training and test sets.

obtained gradients are clipped if they are above a certain threshold since substantial gradient values complicate the learning process and slow it down.

The aim of the trained model is to predict the potential for an extended time period into the future. However, using a minimum squared error loss function for a single prediction, the model's overall performance cannot be estimated decisively. The model can accomplish multiple predictions into the future, but the DL model needs to be interoperable with the PDE solver. Therefore, after every prediction, the following prediction will be affected by feedback provided by the PDE solver, so it is impractical to predict multiple potential values for the next few time steps. In this work, the model's performance evaluation is achieved by comparing the consecutive predictions made by the DL model with validation and test sets. For every five epochs, predictions of the model and the original data are plotted to give a performance estimation as in Figure 5.5. Once a model makes a prediction, the output is added to the last entry of the sequence, and the rest of the entries are shifted one step before. The

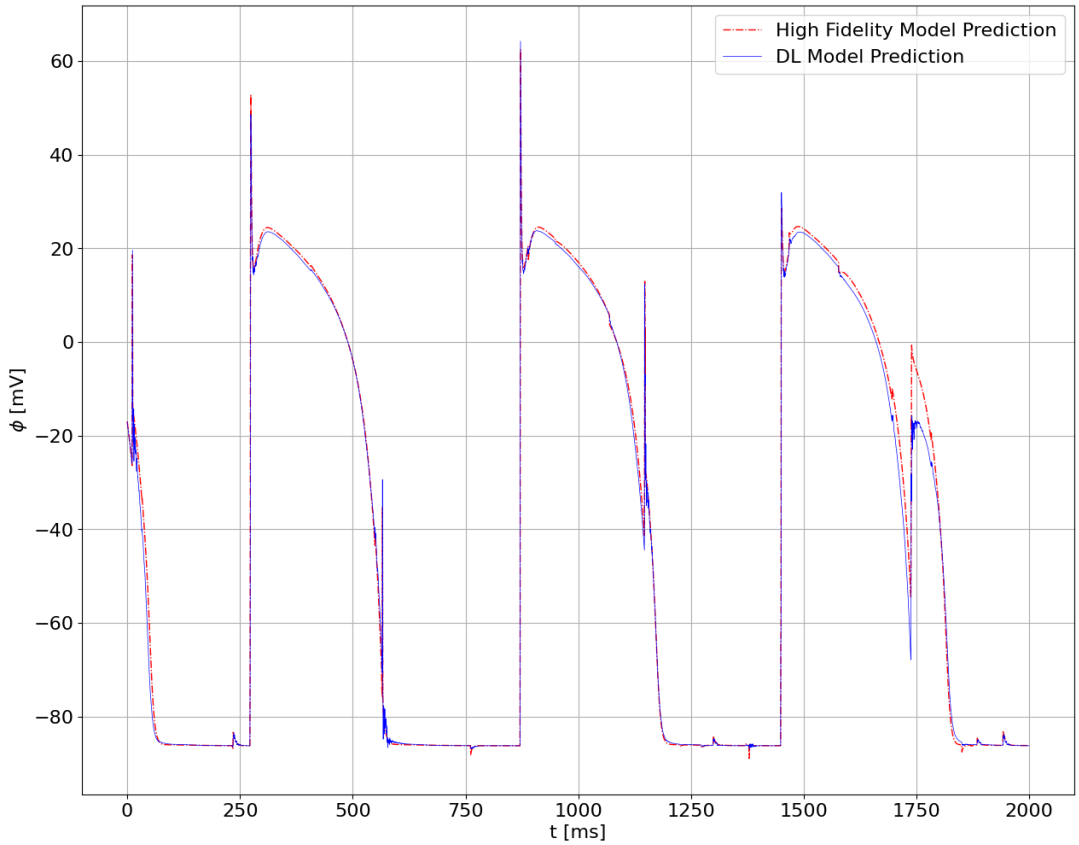

Figure 5.5: Predicted potentials by DL model and potentials from the test set.

stimulus term for the next step is obtained from the data set, and the comparison is completed.

The selection of the DL model is performed based on its loss value. In stochastic gradient-based optimization, the loss values keep fluctuating, and the least loss does not always occur at the last epoch. The loss value is measured in every step of the training, and the prediction performance plot Figure 5.5 is also taken into consideration when saving the model. The model with the minimum loss value and the best performance in the long term is selected and saved as a Torch Script module for the PDE problem.

Figure 5.6 shows the model's predictions over a long time sequence, with stimuli of random magnitudes given with random time intervals. It can be observed that the model demonstrates the classical cardiac cell behavior at the end of effective refractory periods. If the magnitude of the external stimulus is not large enough to make

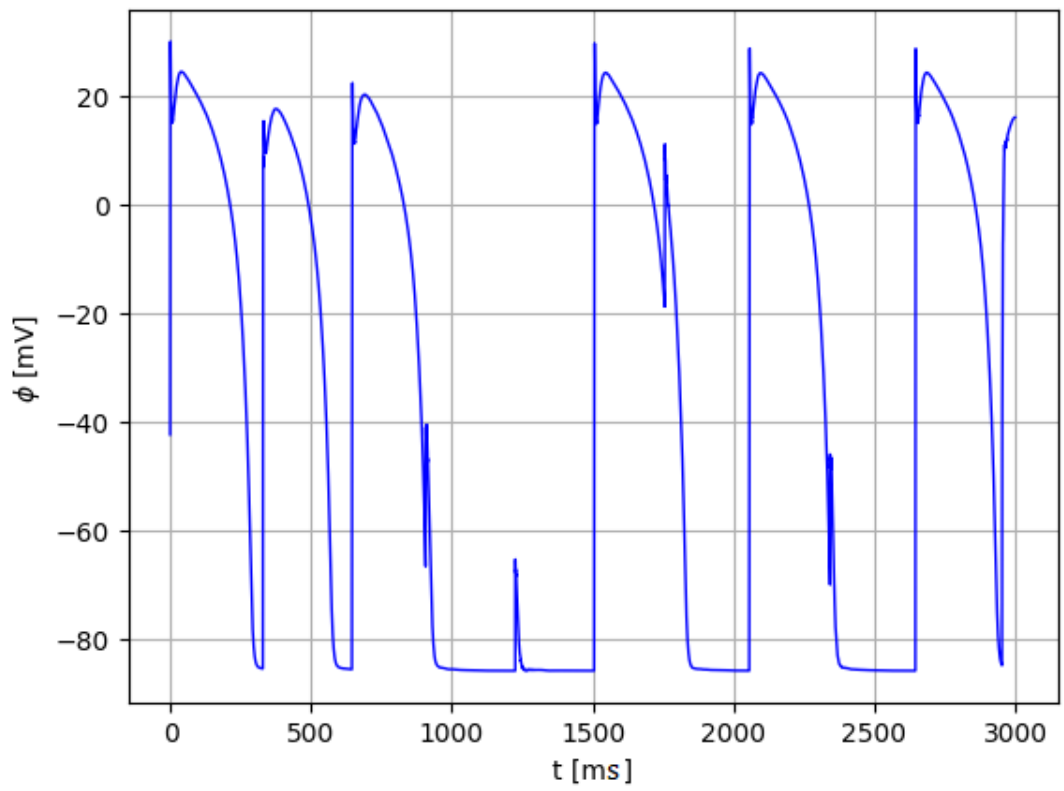

Figure 5.6: DL model prediction over 3000 ms by providing external stimuli.

the potential value pass its threshold, the action potential cycle does not start. Moreover, the action potential duration of the cells changes with the time between action potential cycles and the magnitude of the external stimulus. The DL model is able to show all critical characteristic behavior of a cardiac cell in an ODE setting.

CHAPTER 6

PARTIAL DIFFERENTIAL EQUATIONS AND DEEP LEARNING

The finite element method (FEM) has evolved from a linear structural analysis procedure to a common technique for solving nonlinear, transient PDEs over the last decades. Solving a PDE problem in a complex domain may require enormous time and computational power. Over the years, numerous techniques have been proposed to enhance the performance of the FEM. Wide ranges of PDE solvers specializing in different problems and involving numerical techniques that boost their performance exist today. One of them is the Finite Element Analysis Program (FEAP). FEAP is an academic FEM-based analysis program developed by Professor Robert L. Taylor in the Department of Civil Engineering at UC Berkeley, and it is still actively maintained [81]. FEAP is mainly designed to solve problems in solid mechanics; however, its system can be extended by adding user-developed codes to solve problems in other research areas, such as fluid dynamics and thermoelectric problems. In this study, FEAP is extended to solve the electrophysiological problem of the heart. The operating system where FEAP is used to solve the problem in this study is Linux Ubuntu version 20.04.

6.1 Incorporation of DL Model into FEAP Framework

The proposed DL model is trained with the PyTorch framework. Pytorch is an open-source framework based on the Python programming language and Torch library. Therefore, the model is created with the Python language. The source code of the FEAP is written mainly in Fortran language. To achieve interoperability between the PyTorch model and FEAP, trained PyTorch models are saved as Torch Script models.

LibTorch is a C++ distribution of PyTorch, in LibTorch binary distributions of headers, libraries, and configuration files depending on PyTorch provided by the developers of the PyTorch framework. Saved Torch Script models can be loaded into this C++ API of PyTorch. Loading the model into LibTorch provides a C++ environment, which is more suitable for working with Fortran and C languages. Interoperability between C, C++, and Fortran languages enabled fast and efficient development of algorithms for the focused problem.

For incorporating the DL model into FEAP, mean and standard deviation values of the training data are saved as parameters since the predicted values of the DL model need reverse standardization for rescaling the value back to its original scale, and the feedback from FEAP needs to be standardized to be saved in the history of the DL model. Before deploying the DL model, arrays with shapes matching the input of the DL model are created for each node. Since prior information than the initial condition on the model is generally unavailable, each array filled with resting potentials and no stimuli values as shown,

$$\begin{bmatrix} I_{ext}^{0} = 0 & \phi^{0} = -86.2 \\ I_{ext}^{1} = 0 & \phi^{1} = -86.2 \\ I_{ext}^{2} = 0 & \phi^{2} = -86.2 \\ I_{ext}^{3} = 0 & \phi^{3} = -86.2 \\ \vdots & \vdots \\ I_{ext}^{s-1} = 0 & \phi^{s-1} = \phi_0 \end{bmatrix}, \tag{6.1}$$

where s is the length of sequence, -86.2 is the resting potential in mV, and ϕ_0 is the initial condition of the potential.

One crucial characteristic of this developed method is that the ϕ values can be predicted in the Graphics Processing Unit (GPU). Performing predictions on the GPU relies on the Cuda expansions of the LibTorch framework. DL models' performance is much higher when computations occur in the GPU rather than the Central Process Unit (CPU) due to GPU architecture. However, the memory of GPUs is, in general, more limited, and an array for every node cannot be present in its memory at the same time. During the prediction phase, nodes are loaded in batches to the GPU to navi-

gate this bottleneck. First, a copy of a specified number of history arrays is created in GPU, and then the model outputs are obtained. After that, the next group of history arrays are copied to the GPU. This process is repeated until a prediction is made for all nodes.

Once the stimulation starts, history arrays are initialized for every node, and the coordination between the FEAP and the DL model is achieved by following several steps. First, a ϕ value is predicted with the history data for every node. Then, the FEAP solver computes a general solution by obtaining a source term f^{ϕ} from each node. The source term is calculated as,

$$f^{\phi} = \frac{\phi - \phi_n}{\Delta t} \tag{6.2}$$

through the finite difference method. The predicted f^{ϕ} is assumed to be known priori, and therefore **the linearization of the problem** is achieved. The linearization of the problem means the model converges to a solution after a single calculation step. Once the FEAP computes the solution by using the corrected potentials ϕ_c resulting from the FEM solution, the stimulus term I_{ext} in the last row of the history arrays is calculated using the finite difference method,

$$I_{ext} = \frac{\phi_c - \phi_p}{\Delta t} \tag{6.3}$$

and corrected in the history array. The assumption is that the difference between corrected ϕ_c and predicted ϕ_p potential values is caused by only the PDE's divergence term div$\mathbf{q}$. The effects of this divergence term on the source term are taken into account by DL in the next prediction step. Therefore, the effects of divergence term on the source terms are delayed by a single time step. Following this algorithm, the divergence term of the PDE is associated with the stimulus term used in the training of the DL model. It should be noted that the variables taken from the history array and the FEAP solution must be on the same scale, and standardization and reverse standardization are applied to the history variables in the given equations. The proposed algorithm can be demonstrated in the following form,

Prediction Stage

$$\begin{bmatrix} I_{ext,0} & \phi_0 \\ I_{ext,1} & \phi_1 \\ I_{ext,2} & \phi_2 \\ \vdots & \vdots \\ I_{ext,n} & \phi_n \end{bmatrix} \xrightarrow{Prediction} \phi_{p,n+1} \xrightarrow{f^{\phi}\ Calculation} \frac{\phi_{p,n+1} - \phi_n}{\Delta t} = f^{\phi}$$

Correction Stage

Step 1: $FEAP(\text{div}\mathbf{q}, External\ Flux, f^{\phi}) \xrightarrow{Correction} \phi_{c,n+1}$

Step 2: $\phi_{c,n+1} \xrightarrow{I_{ext}\ Correction} \frac{phi_{c,n+1} - phi_{p,n+1}}{\Delta t} = I_{ext,n+1}$ (6.4)

Updated History : $\begin{bmatrix} I_{ext,1} & \phi_1 \\ I_{ext,2} & \phi_2 \\ I_{ext,3} & \phi_3 \\ \vdots & \vdots \\ I_{ext,n+1} & \phi_{n+1} \end{bmatrix}$.

The $FEAP(\text{div}\mathbf{q}, External\ Flux, f^{\phi})$ term demonstrates that FEAP takes external flux and divergence terms into account with the source term.

Two main reasons exist for directly utilizing ϕ values in the history arrays instead of f^{ϕ} values. Correcting f^{ϕ} values directly from the FEAP solver is more complex and requires checking the potential history on that node since divergence is not a direct output. Also, if a model is trained with f^{ϕ} data instead of ϕ, it would fail in PDE without knowledge from the primary ϕ field. Utilizing both f^{ϕ} and ϕ in the history array is observed to be inefficient and has high computational costs. It is generally easier for a model to learn normally distributed data. Even though the distribution of potential ϕ values is very distinct from normal distribution, as can be seen from Figure 6.1, they are still more structured and easy for the DL model to learn than the f^{ϕ} values. Figure 6.2 demonstrates the ϕ and f^{ϕ} values for the same action potential cycle.

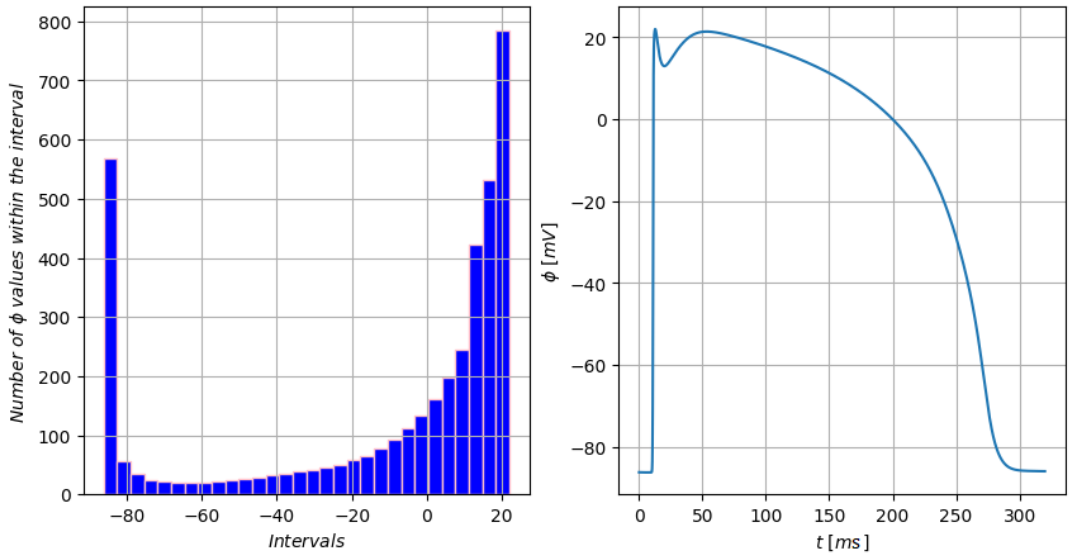

Figure 6.1: Histogram of potential ϕ distribution in a complete action potential cycle.

It can be observed that extreme outliers are present in the f^{ϕ} distribution with critical importance on characteristics of action potential cycle, and training an NN for this data is significantly more challenging.

6.2 Lumping in PDE

In the context of FEM, lumping refers to a computational technique used in matrices and vectors arising from the PDE domain's discretization. The computational efficiency of the cardiac electrophysiology problem is examined in [71]. In this study, different combinations of lumped approximations are studied and shown to be effective in the mono-domain problem of cardiac electrophysiology. The lumping technique's primary purpose is to simplify the computations of matrices and vectors.

For example, in a mass matrix, element masses are redistributed onto the diagonal of the matrix, and off-diagonal terms are neglected. This process results in a diagonal matrix where each diagonal entry represents the lumped term associated with the corresponding degree of freedom. Since lumped matrices are diagonal, they are computationally efficient to solve and can take less memory to store. This technique is commonly utilized in solving dynamic problems and large-scale problems. However,

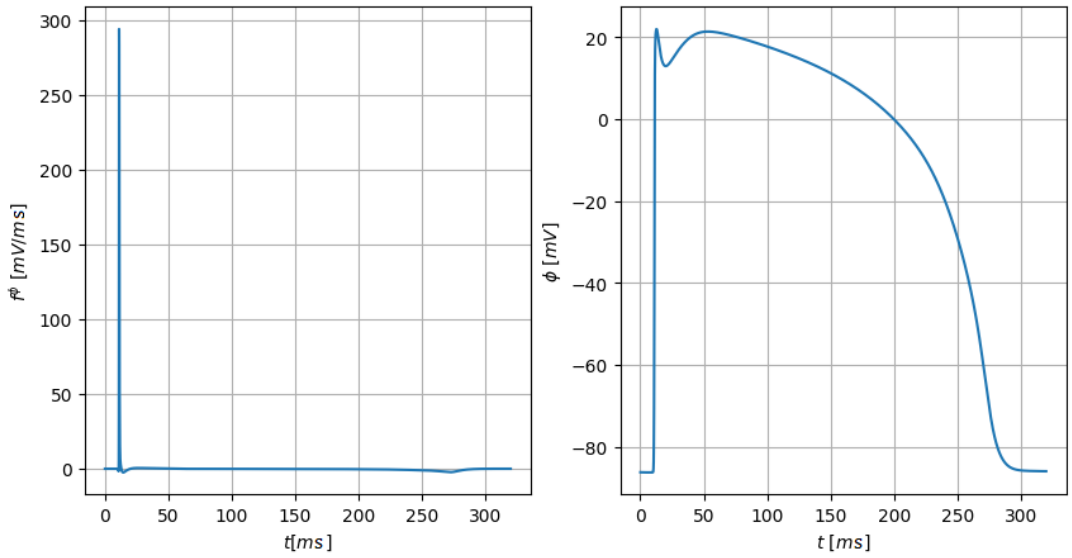

Figure 6.2: f^ϕ and ϕ changes in a single action potential cycle.

lumping causes a loss of accuracy since the off-diagonal coupling between terms is disrupted with this method. Therefore, it offers a trade-off between accuracy and computational efficiency.

Holding the history arrays at integration points in the Gauss quadrature method can introduce artificial stimuli when f^ϕ values are projected to the nodes. Therefore, lumping can significantly improve the proposed predictor-corrector algorithm with increased computational efficiency. The main equation of the PDE can be formulated as,

$$\dot{\phi} - \text{div}\mathbf{q} - f^\phi = 0 \tag{6.5}$$

with ignoring the external flux term for simplification. Equation (6.5) can be cast into the residual form for each element in the following FEM form,

$$\mathbf{r}_e = \int_{\mathcal{B}_e} \mathbf{N}^T\mathbf{N}\frac{[\Phi_e - \Phi_e^n]}{\Delta t}dV + \int_{\mathcal{B}_e} \mathbf{B}^T\mathbf{q}dV - \int_{\mathcal{B}_e} \mathbf{N}^T\mathbf{N}\mathbf{f}_e^{\Phi}dV,$$

$$\mathbf{k}_e = d_{\Phi_e} r_e = \int_{\mathcal{B}_e} \mathbf{N}^T\mathbf{N}\frac{1}{\Delta t}dV + \int_{\mathcal{B}_e} \mathbf{B}^T d_{\nabla\Phi^e}\mathbf{q}\mathbf{B}dV - \int_{\mathcal{B}_e} \mathbf{N}^T\mathbf{N}d_{\Phi_e}\mathbf{f}_e^{\Phi}dV, \quad (6.6)$$

where $\mathbf{m}^e = \mathbf{N}^T\mathbf{N}$ & $\mathbf{k}^e = \mathbf{B}^T\mathbf{B}$.

In equation (6.6), shape functions are denoted with $\mathbf{N}$, and the gradient of the shape functions are denoted as $\mathbf{B}$. In the proposed approach, only the mass matrix terms $\mathbf{m}^e$ are lumped, and the conductivity matrix $\mathbf{k}^e$ is kept unaltered.

CHAPTER 7

RESULTS

This chapter examines the performance of the DL model with the proposed algorithm under different conditions. Section 7.1 compares the effects of lumping to the non-lumped approach in the 2D domain. Section 7.2 compares the general behavior of the proposed DL-based algorithm with the ten Tusscher-Panfilov model-based algorithm in different cases. In Section 7.3, a realistic 2D ventricular model is used for a PDE solution, and results are demonstrated. In Section 7.4, the complex reentry behavior that can occur in cardiac electrophysiology is created and solved with the DL-based algorithm.

The simulations are performed in a Dell Inspiron machine with 16 gigabytes of DDR4 RAM and an NVIDIA GeForce GTX 1050 Ti GPU with 4 gigabytes of memory. The device's processor has four cores and a 2.8 gigahertz process frequency.

7.1 Effects of Lumping

The proposed algorithm is modified to a default non-lumped form. The history arrays are still kept at the nodes of the elements; however, the returned source values to the FEAP solver are obtained by interpolating the source values at Gauss quadrature points, and lumping is not applied to any matrix or vector.

Figure 7.1 shows the state of potentials at different time steps of the solution. The same DL model trained with $\Delta t = 0.4$ ms time steps was used in both solutions. The considered domain is 50×50 mm^2 homogeneous cardiac tissue block discretized into 60×60 four-node quadrilateral elements. The conduction tensor is assumed

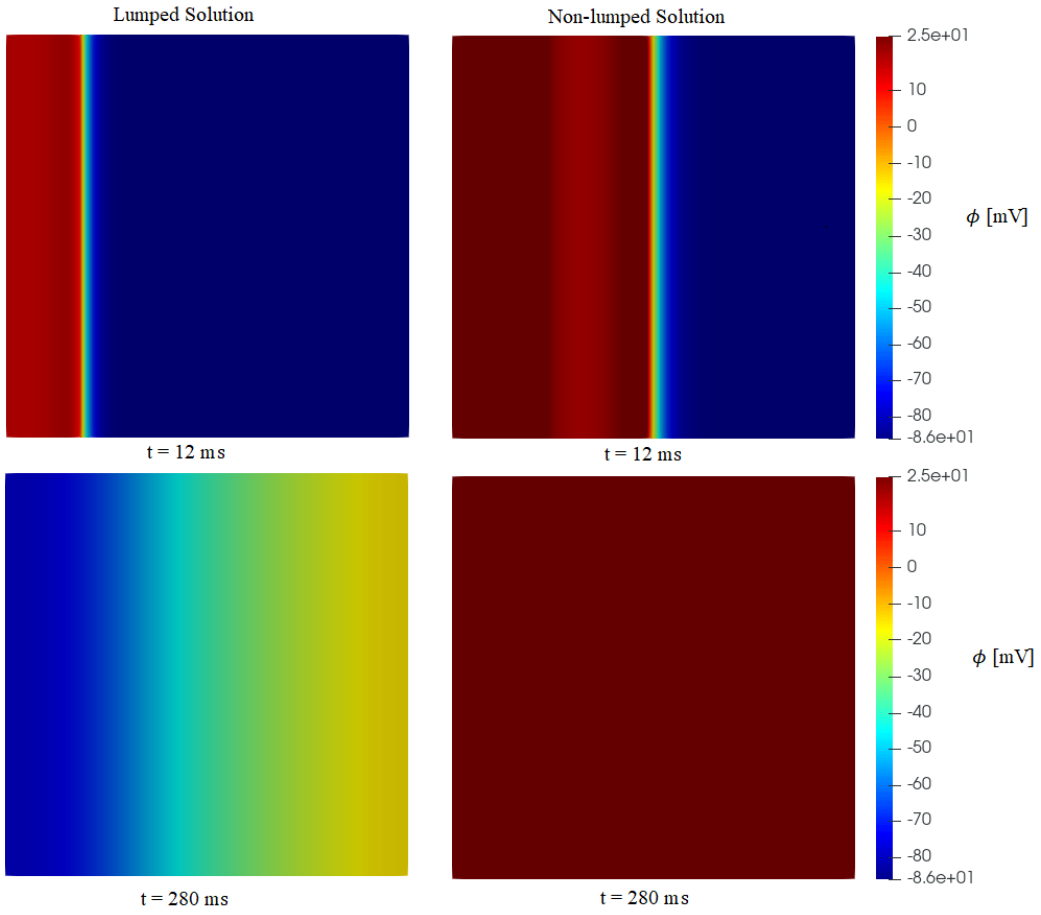

Figure 7.1: The state of potential ϕ values over the domain, provided for lumped and non-lumped approaches at the solution time of $t = 12$ ms and $t = 280$ ms.

to be isotropic with $\mathbf{D} = d^{iso}\mathbf{I}$ with $d^{iso} = 1.0$ mm/ms. An external flux of 120 mV/ms is applied to the nodes at the left edge of the domain, towards the center of the domain, for 0.8 ms. Applied external fluxes managed to excite the cells successfully. The wavefront moved through the entire domain in both lumped and non-lumped versions. Around $t = 280$ ms, the repolarization of the problem being solved with the lumped version started, while the non-lumped version still had extremely high potential values all over the domain.

Considering the source terms at the Gauss integration points instead of the element nodes as in the non-lumped version generated extreme stimuli values, and the DL model could not predict correct source values. Figure 7.2 shows the potential changes at the central nodes of both domains with respect to time. Even though the depolar-

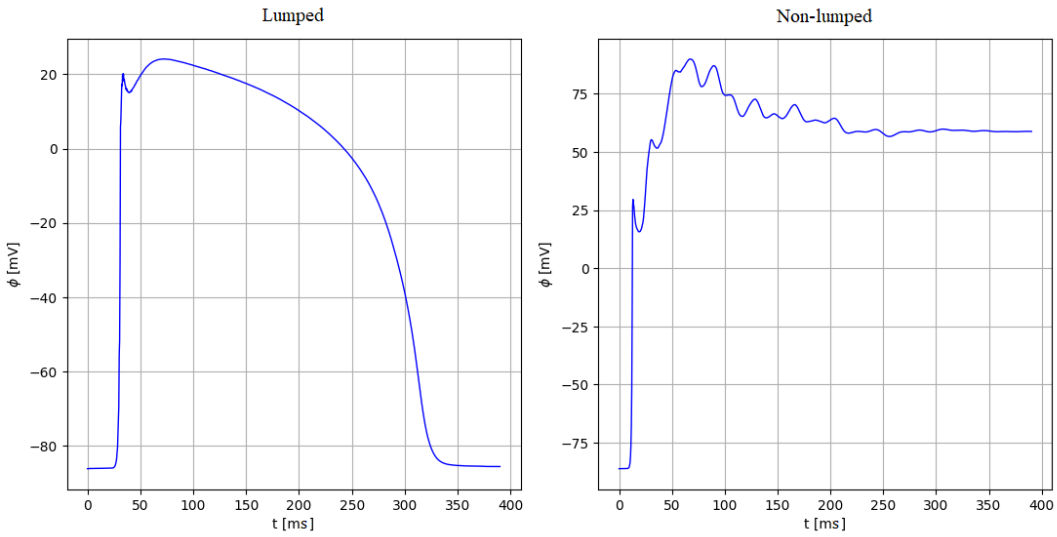

Figure 7.2: Potential values with respect to time from the central point from lumped and non-lumped solution.

ization and initial repolarization are reflected in the non-lumped version, extreme stimulus terms prevent the model from repolarizing, while in the lumped version, the same node repolarized successfully and favorably.

7.2 Deep Learning and High Fidelity Model-Based Solution Comparison

Comparison between the solution provided by the DL-based algorithm and high fidelity ten Tusscher-Panfilov model is examined in several cases.

The results of a DL model, which is trained with time steps of $\Delta t = 0.4$ ms, is compared to the results of ten Tusscher-Panfilov model with time steps of $\Delta t = 0.08$ ms in an ODE setting. At every 350 ms, an influx of 120 mV/ms is applied to the point for a duration of 0.8 ms. Figure 7.3 demonstrates the results of both approaches. DL-model shows extremely great accuracy even for an extended period of time of 5000 ms. There is an overshooting at the depolarization stage of the DL model. However, the model successfully follows its path and readjusts itself accordingly.

A comparison between the outcomes of the DL-based algorithm and the high fidelity model-based algorithm using the same time steps is demonstrated in Figure 7.4. An

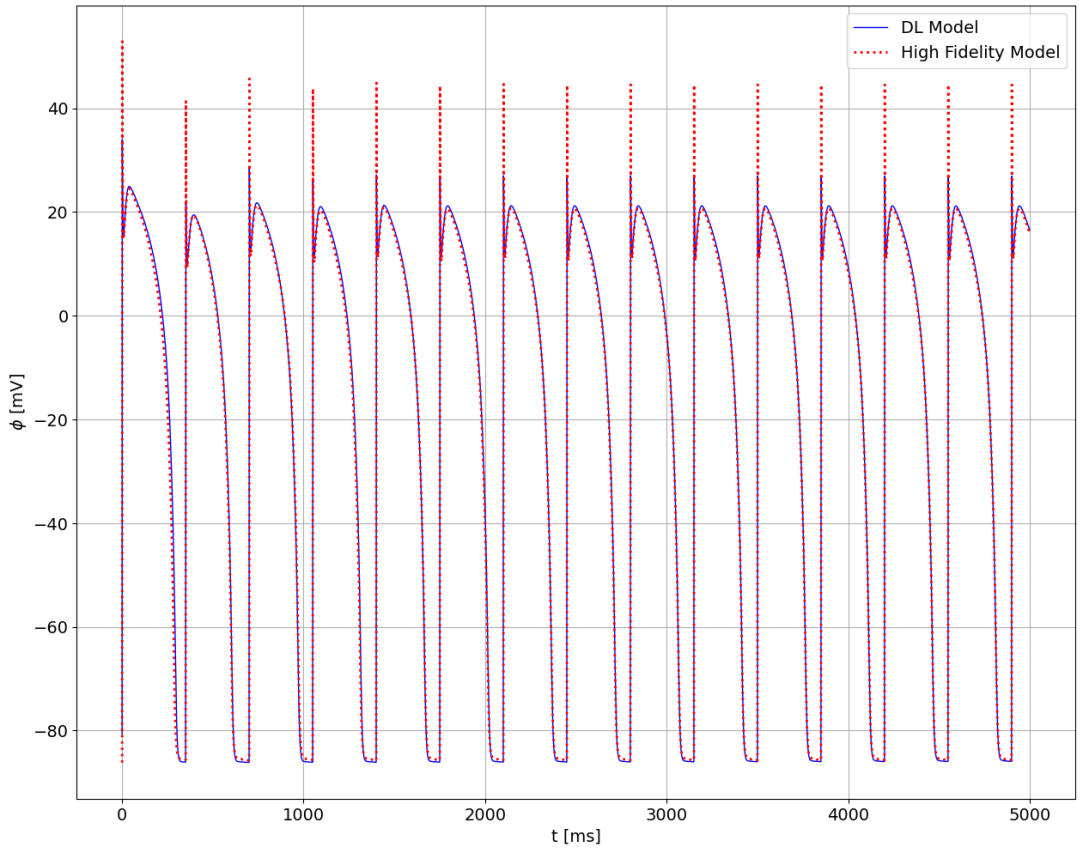

Figure 7.3: Potential values predicted by the DL model and the solution of high fidelity model as ODE problems over time $t = 5000$ ms.

additional pooling layer is added after the last convolutional layer of the DL model, and the sequence length of the input array is doubled to learn the action potential cycle with time steps of $\Delta t = 0.2$ ms since it is essential for the DL model to know about a large segment of the wave prior making a prediction. The considered domain is 50×50 mm^2 homogeneous cardiac tissue block discretized into 100×100 four-node quadrilateral elements. The conduction parameter is considered $d^{iso} = 3.0$ mm/ms, and only isotropic conduction is present. An external flux of 120 mV/ms is applied to the nodes on the left edge of the domain, towards the center of the domain, for 0.8 ms. The wavefront moves faster in the high fidelity model-based solution than the DL-based one. As a result, the repolarization of the high fidelity model-based algorithm has happened before.

The difference between the APD of the solutions is around 20 ms based on Figure 7.5. Overall, the action potential cycle is similar, and the results of the DL-based

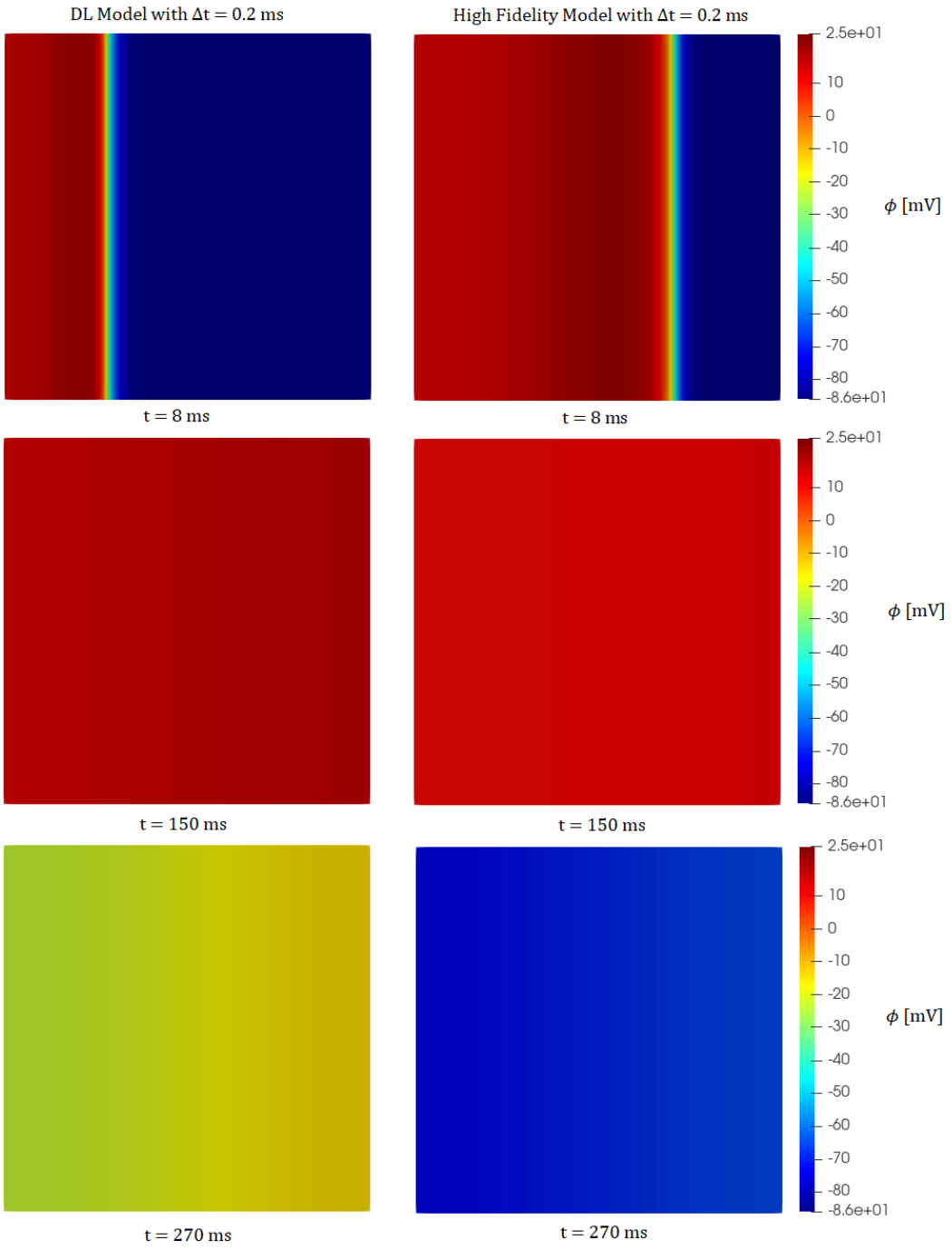

Figure 7.4: Snapshots from the solutions obtained with a DL-based algorithm trained with $\Delta t = 0.2$ and a high fidelity model with time steps of $\Delta t = 0.2$.

algorithm are acceptable for many applications.

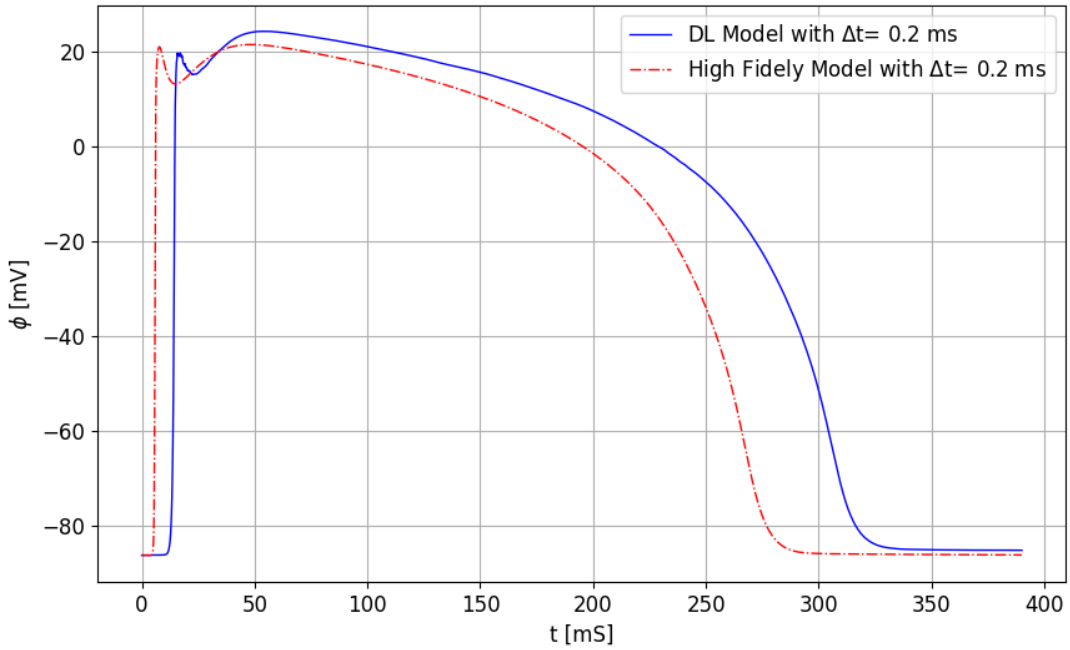

Figure 7.5: Potential values with respect to time from the central point from DL and high fidelity model-based solutions.

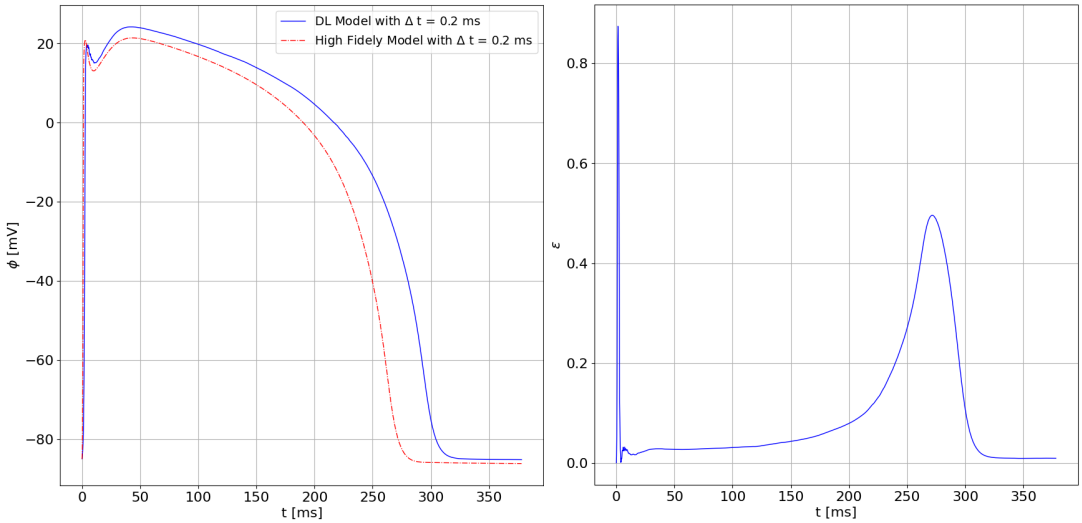

Figure 7.6: Demonstration of action potential cycles for DL-based and high fidelity-based solutions shifted to t=0 and obtained error values.

The error is defined as

$$\epsilon = \frac{(|\phi_{DL} - \phi_{HF}|)}{100}. \tag{7.1}$$

In error definition, ϕ_{DL} is the potential value at a given time obtained from a DL-

based solution, ϕ_{HF} is the potential value at a given time obtained from a solution with the high fidelity model, and 100 is a factor to normalize the error. In the rest of this study, errors are calculated using the Equation (7.1), and the high fidelity model is accepted as the ground truth. Figure 7.6 demonstrates the difference between the solutions obtained from the high fidelity model and the DL-based model when the start of the depolarization occurs at the same time. It can be observed that after the initial depolarization stage, the error reaches acceptable levels. However, due to the delay in the repolarization stage of the DL-based approach, the error between the solutions increases to around $\epsilon = 0.45$. After the repolarization stage is complete, both approaches reach the resting potential value, and the error is around $\epsilon = 0.01$. The simulation times are around 44 minutes for the DL-based algorithm and 18 minutes for the high fidelity model-based algorithm. It should be taken into consideration that the time step of $\Delta t = 0.2$ ms is close to the maximum allowed step size by the high fidelity model and generally is not used in the simulations. In contrast, this time step does not cause significant problems for the DL model and can be increased without introducing convergence issues.

For high fidelity ten Tusscher-Panfilov, the step used in the original publication is $\Delta t = 0.02$ ms [13]. Two DL-based algorithms with time steps of $\Delta t = 0.2$ ms and $\Delta t = 0.4$ ms are compared to the high fidelity model-based solution. The high fidelity model-based algorithm is considered the ground truth. The considered domain is 50×50 mm^2 homogeneous cardiac tissue block discretized into 60×60 four-node quadrilateral elements. The conduction is assumed to be isotropic with $d^{iso} = 1.0$ mm/ms. An external flux of 120 mV/ms is applied on the nodes at the left edge of the domain, towards the center of the domain for 0.8 ms. Wavefront in the ten Tusscher-Panfilov model-based stimulation moves faster than in the DL-based simulations. All three solutions excite the whole domain and repolarize around 300 ms. Accuracy and the conduction of both DL-based simulations are very similar. The proposed algorithm with the DL model trained with time steps of $\Delta t = 0.4$ ms solves the problem in 8 minutes, the model trained with time steps of $\Delta t = 0.2$ ms solves the problem in 15 minutes, and the high fidelity model solves the problem in 61 minutes.

The overall action potential behavior is similar in both DL model simulations, as shown in Figure 7.8. Error analysis of the DL model trained with different time

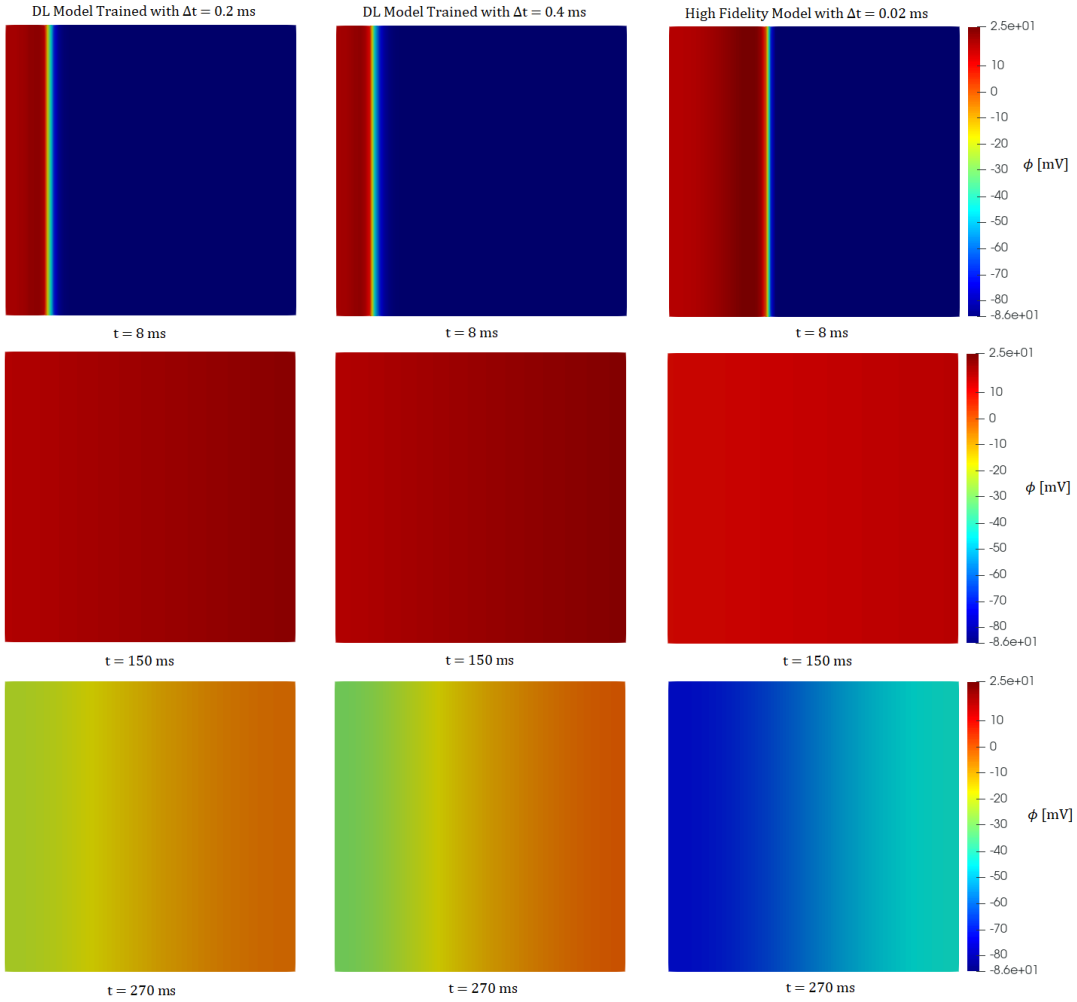

Figure 7.7: Snapshots from the solution obtained by DL-based algorithm with time steps of $\Delta t = 0.2$ ms and $\Delta t = 0.4$ ms, and high fidelity model-based algorithm with a time step of $\Delta t = 0.02$ ms.

steps is presented in Figure 7.9. The APD of the DL simulations is around 25 ms longer than the APD of the high fidelity model. APD of the DL stimulation with the model trained with $\Delta t = 0.4$ ms is around 5 ms shorter than the model trained with $\Delta t = 0.2$ ms. The error increases to $\epsilon = 0.4$ at the end of the repolarization phase for both DL-based solutions due to increased action potential duration. At the end of the repolarization phase, both DL-based solutions reach the resting potential, and error is minimized. Overall, both DL models offer adequate accuracy.

Nonplanar excitation of the domain requires complex computations. Figure 7.10 shows nonplanar wave propagation in the domain simulated by a DL-based algo-

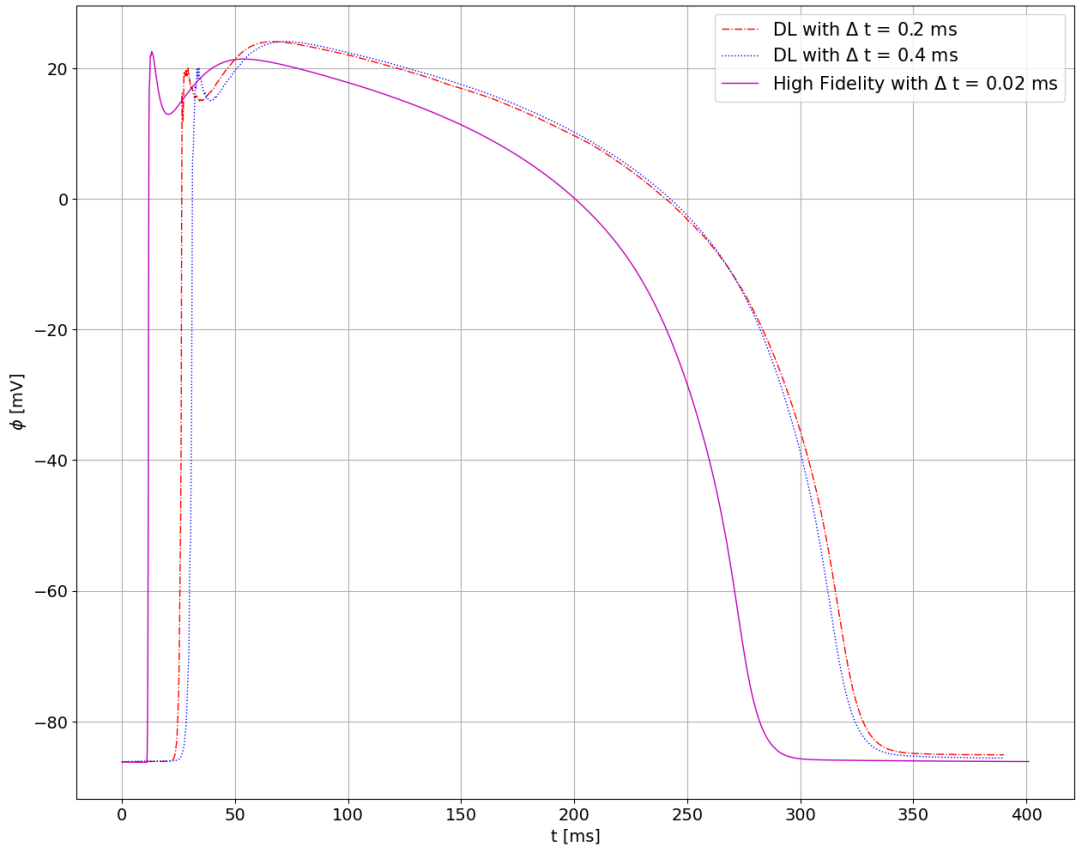

Figure 7.8: Potential values with respect to time from the central point from DL and High Fidelity Model simulations demonstrated in 7.7.

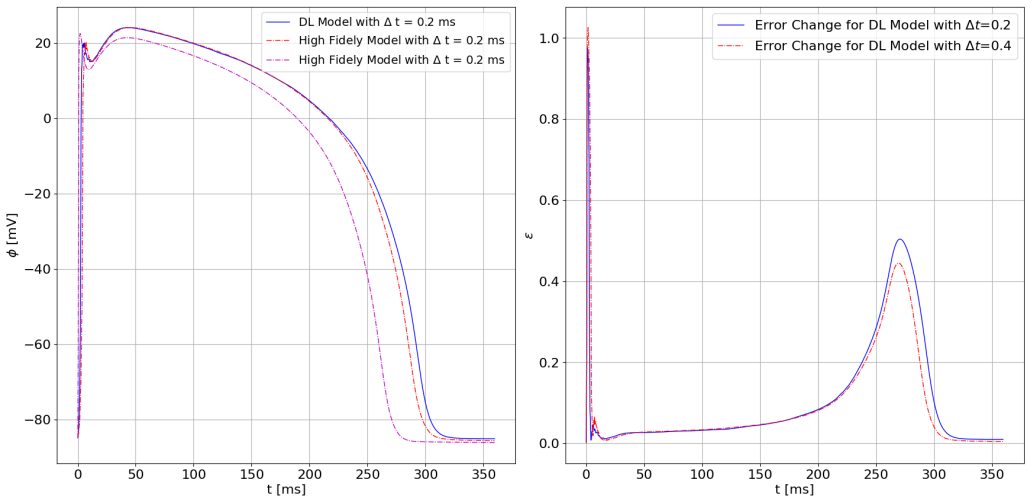

Figure 7.9: Demonstration of action potential cycles for DL-based and high fidelity solution shifted to t=0 and obtained error values based on 7.7.

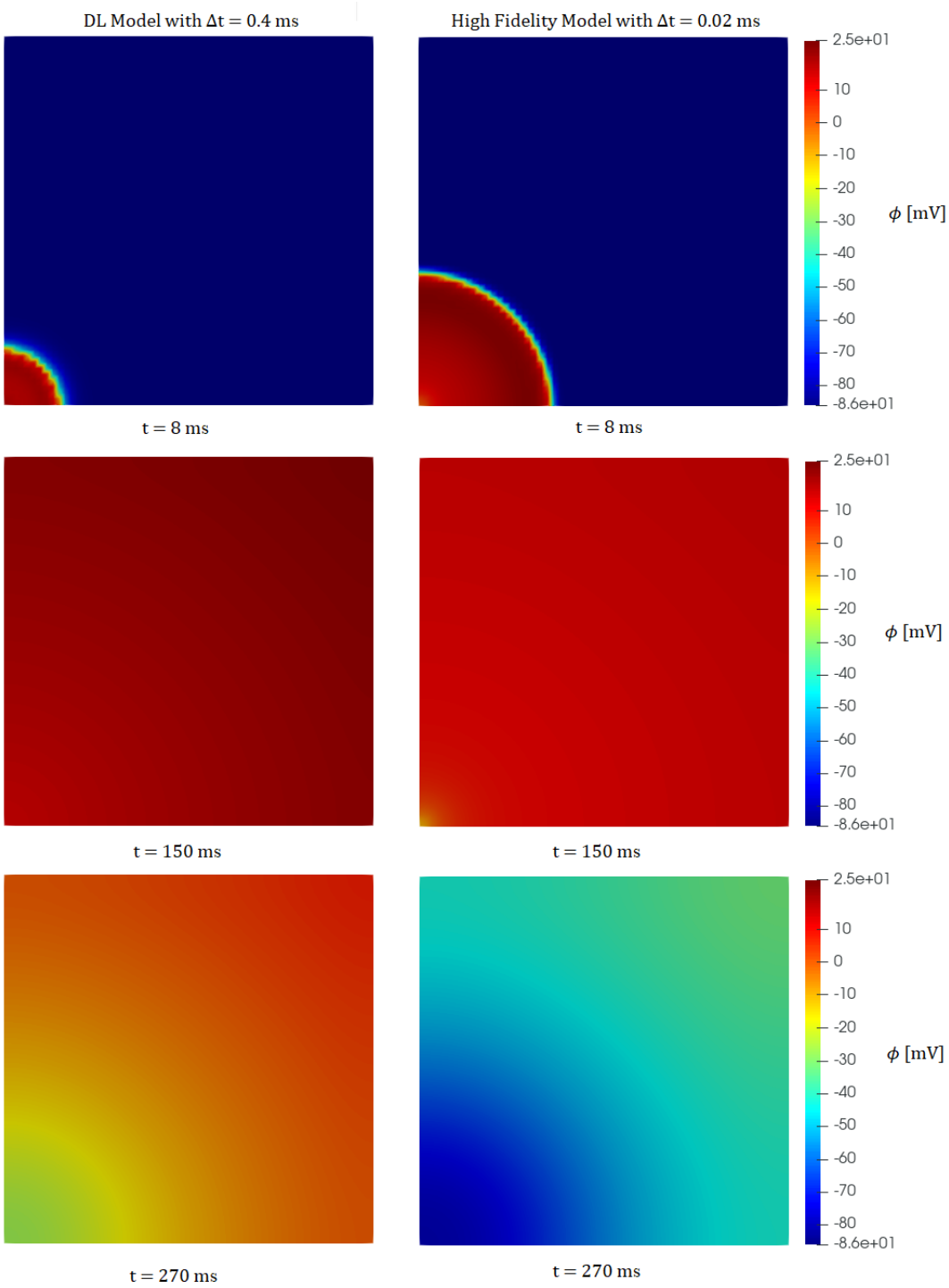

Figure 7.10: Snapshots from the solution obtained by nonplanar excitation of the domain with DL-based algorithm with a time step of $\Delta t = 0.4$ ms and high fidelity model-based algorithm with a time step of $\Delta t = 0.02$ ms.

rithm with a time step of $\Delta t = 0.4$ ms and a high fidelity model-based algorithm with a time step of $\Delta t = 0.02$ ms. The considered domain is 50×50 mm^2 homogeneous cardiac tissue block discretized into 60×60 four-node quadrilateral elements. The conduction is assumed to be isotropic with $d^{iso} = 1.0$ mm/ms. An external flux of 120 mV/ms is applied to the 9 nodes at the left bottom corner towards the right edge of the domain for 0.8 ms. DL-based simulation has managed to capture the correct wavefront propagation, with a lower wavefront speed than the high fidelity ten Tusscher-Panfilov model with $\Delta t = 0.02$ ms.

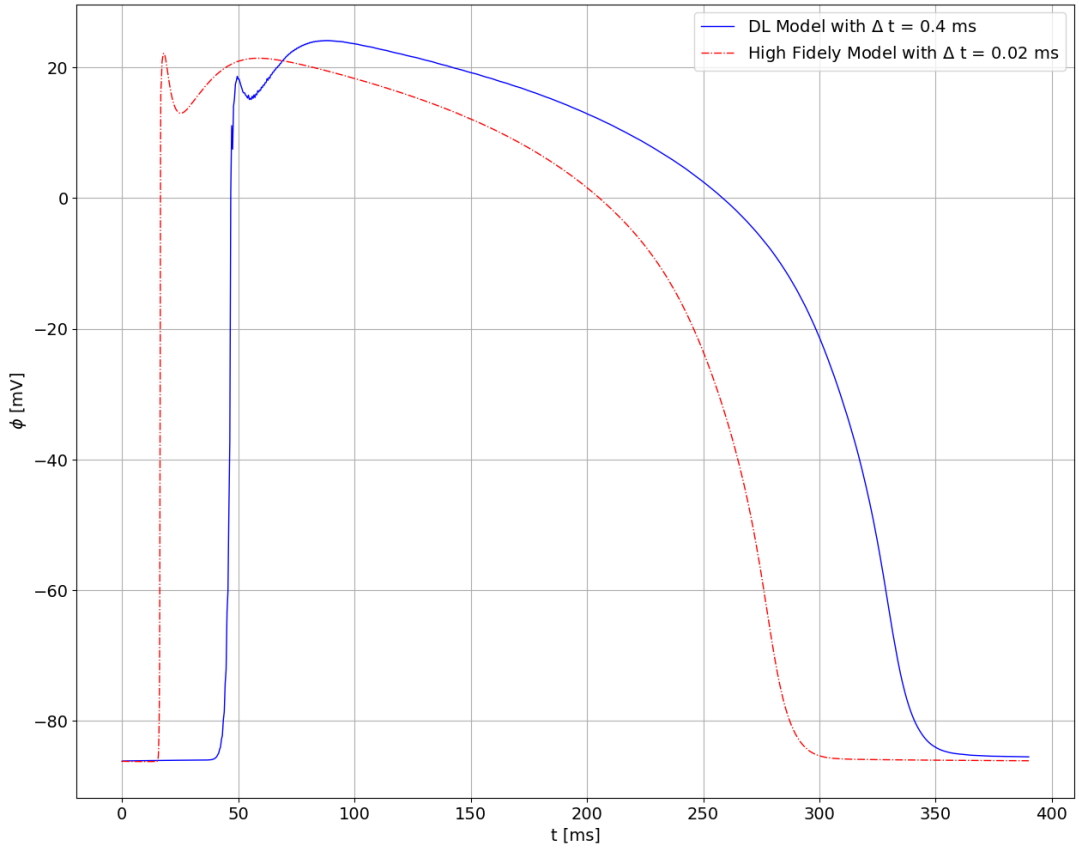

Figure 7.11: Potential values with respect to time from the central point from DL and High Fidelity Model simulations demonstrated in 7.10.

The shift in the depolarization stage is observable between the nonplanar wave propagation cases presented in Figure 7.11. The wavefront in the high fidelity solution reaches the central point in the domain around 25 ms earlier than the DL-based solution. Moreover, the action potential duration of the DL-based solution is around 30 ms longer than the high fidelity model solution. After the repolarization is complete,

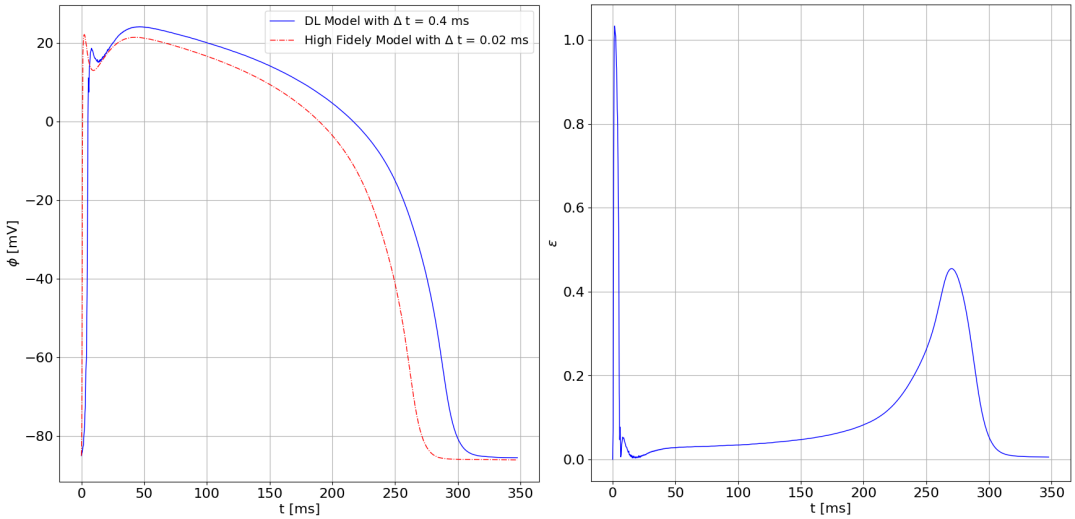

Figure 7.12: Demonstration of action potential cycles for DL-based and high fidelity solution shifted to t=0 and obtained error values based on 7.10.

both solutions stay around the resting potential value, and the error is minimal.

7.3 DL-based Algorithm Solution in Realistic Ventricular Geometry

The complex geometry of the heart's ventricles increases the problem's complexity. Correct modeling of the ventricles opens the possibility of observing the mechanical behavior of the heart and creating artificial ECG from the simulations.

Figure 7.13 demonstrates the wavefront of action potential in ventricles and potential values over the domain. Since the Purkinje fibers are not present in this model, the wave propagation is expected to move from the upper septum to the apex of the heart and then to the ventricle walls. The considered ventricular tissue block has 1939 four-node quadrilateral elements. The conduction is assumed to be isotropic with $d^{iso} = 1.0$ mm/ms. An external flux is applied to the nodes at the top of the septum, with 70 mV/ms in the horizontal direction towards the right ventricular wall and 70 mV/ms in the vertical direction towards the apex for 0.8 ms. The DL-based algorithm successfully solves the electrophysiological problem of the cardiac ventricles around 18 minutes. The time needed for ten Tusscher-Panfilov model with time steps of $\Delta t = 0.08$ ms to solve the same problem is around 14 minutes, and for time steps

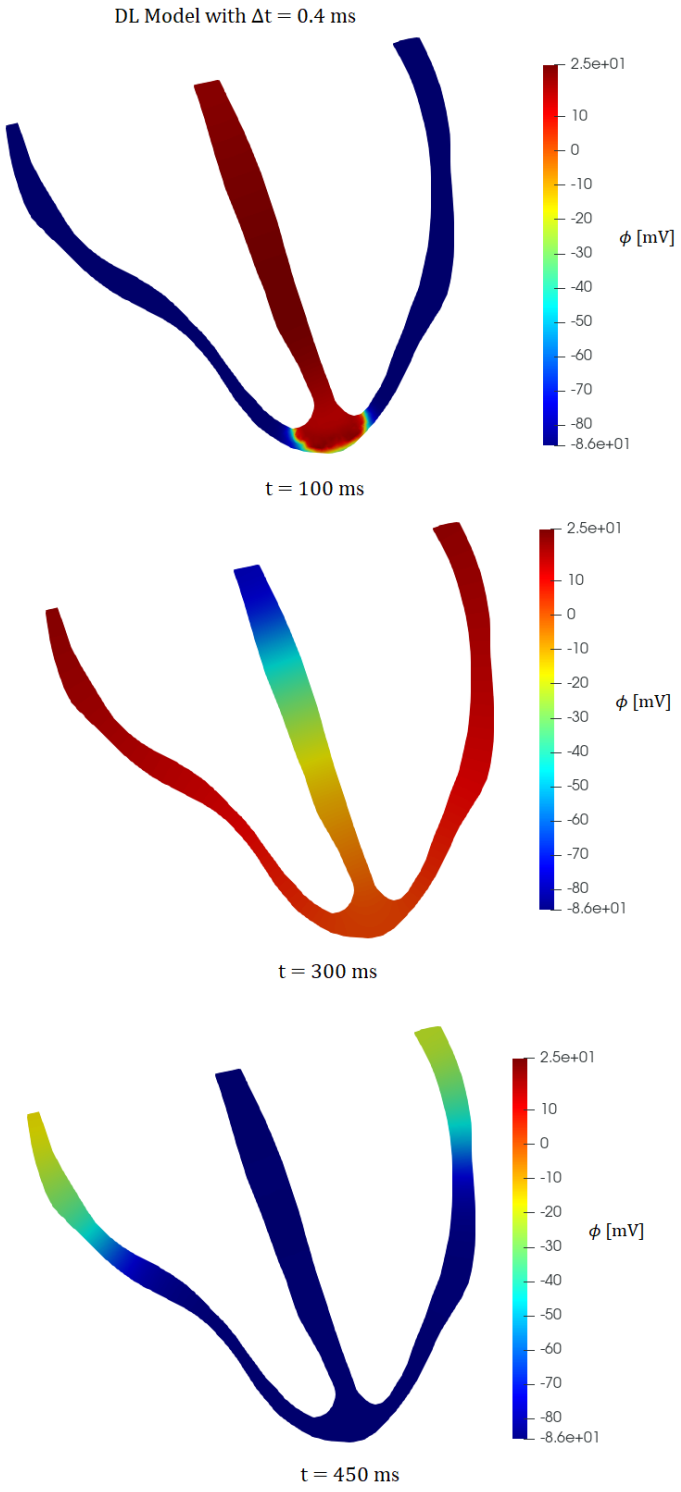

Figure 7.13: Snapshots from the solution obtained with the DL-based algorithm with the time step of $\Delta t = 0.4$ ms in ventricular geometry.

of $\Delta t = 0.02$, which is the recommended time step size, the problem is solved in 53 minutes.

The error analysis for wave propagation in ventricular geometry is performed by computing a heart vector $\mathbf{q}^{heart}$ [82]. The vector $\mathbf{q}^{heart}$ is the integrated sum of the electrical flux $\mathbf{q}$ at all points in the domain. The vector $\mathbf{q}^{heart}$ can be calculated as,

$$\mathbf{q}^{heart} = \int_{\mathcal{B}} \mathbf{q} dV \text{ with } \mathbf{q} = \mathbf{D}\nabla\phi. \tag{7.2}$$

Since an implicit finite element approach is applied in the solution, potential flux $\mathbf{q}$ terms are numerically integrated with the Gauss quadrature method to obtain $\mathbf{q}^{heart}$ vector.

In Figure 7.14, the first and second components of the vector $\mathbf{q}^{heart}$ are demonstrated. Even though both DL-based and high fidelity model-based solutions demonstrated similar overall behavior, the delay in the DL-based solution is a reason for large errors.

In Figure 7.15, components of the $\mathbf{q}^{heart}$ vector obtained from the DL-based solution are shifted in time to remove the delay. However, it can be observed that the evolution of the components obtained from the DL-based solution is significantly slower than those obtained from high fidelity model-based solution, and more delay occurs. Changes in the components of $\mathbf{q}^{heart}$ occur in larger time intervals in the results of the DL-based framework. Another observation is that the magnitude scale of both components is in a similar range in both solutions.

7.4 DL-based Algorithm Solution for the Reentry Behavior

Reentry is a challenging phenomenon to model in a computational environment. It is critically important to model reentries with the DL-based algorithm to capture different pathological conditions that cause cardiac arrhythmia.

Figure 7.16 shows the change in potential across the cardiac tissue, imposed reentry conditions, and reentry behavior. The considered domain is 50×50 mm^2 homogeneous cardiac tissue block discretized into 60×60 four-node quadrilateral elements.

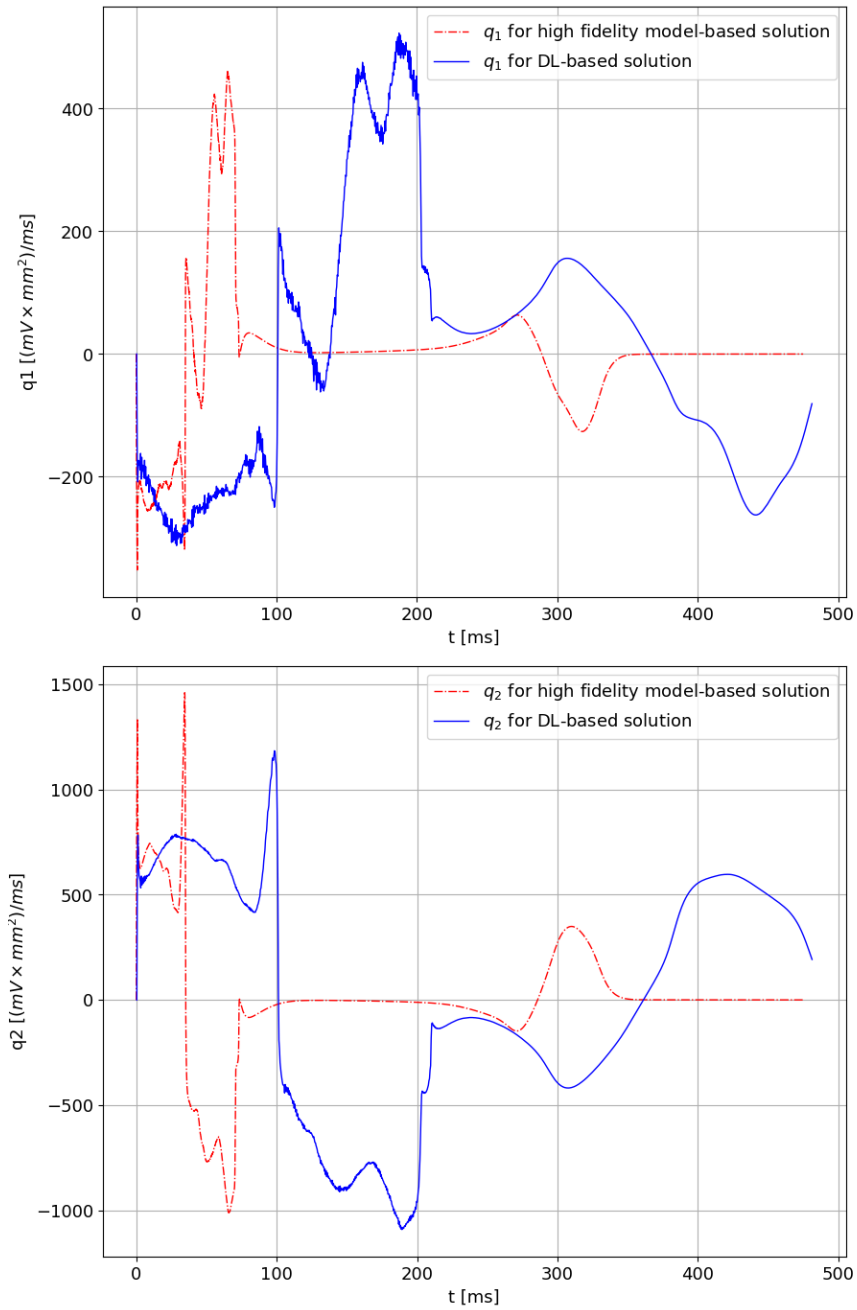

Figure 7.14: Change of first and second component of $\mathbf{q}^{heart}$ with respect to time for DL-based solution with $\Delta t = 0.4$ ms and high fidelity model-based solution with $\Delta t = 0.08$.

The conduction is assumed to be isotropic with $d^{iso} = 1.0$ mm/ms. An external flux of 120 mV/ms is applied to the nodes at the left edge of the domain, towards the center of the domain, for 0.8 ms. In addition, between the time $t = 318$ ms and $t = 328$ ms,

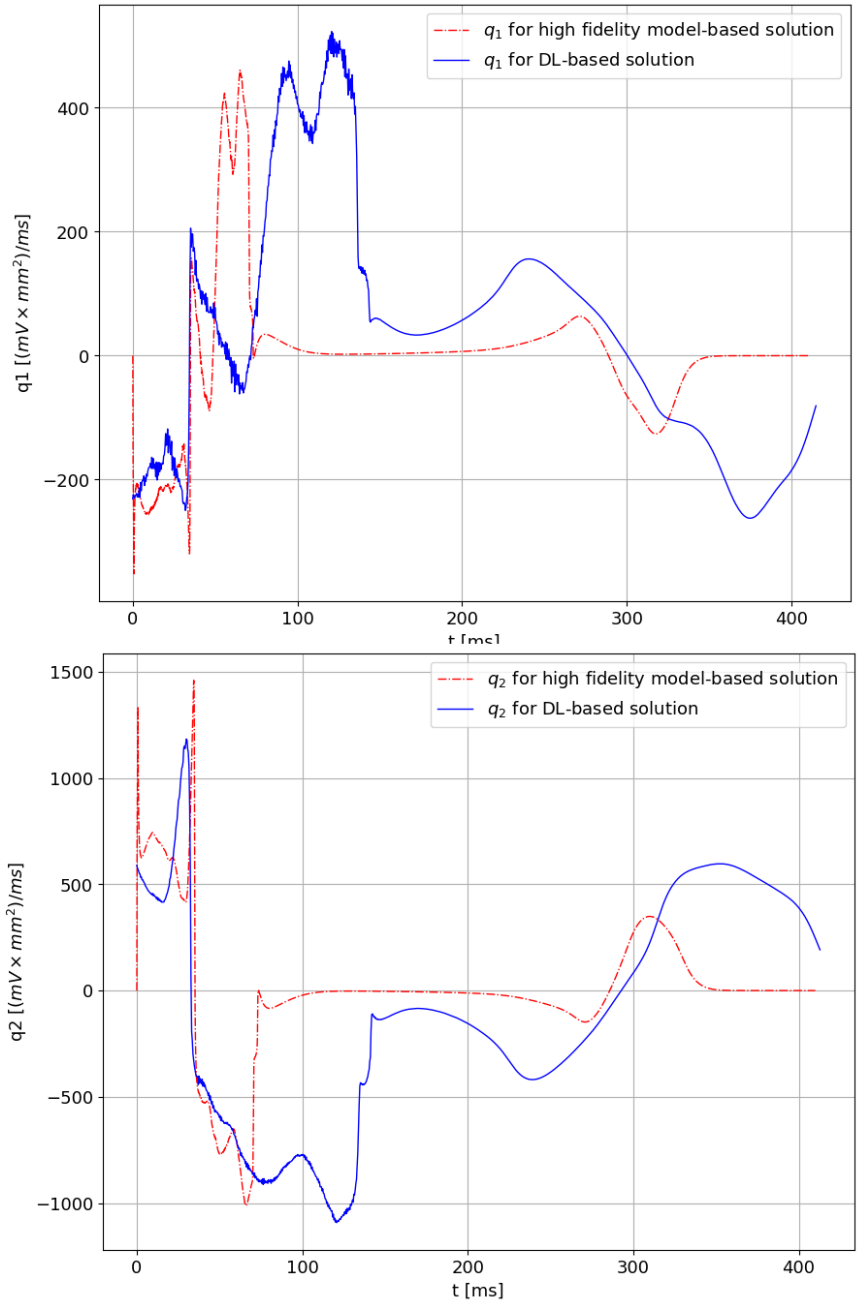

Figure 7.15: Shifted version first and second component of $\mathbf{q}^{heart}$ with respect to time for DL-based solution with $\Delta t = 0.4$ ms and high fidelity model-based solution with $\Delta t = 0.08$.

a flux of 50 mV/ms in the direction of the right edge of the domain is given to a block of nodes below the central point of the domain. The second flux is given to initiate the reentry behavior at the end of the effective refractory period. The proposed DL-based algorithm successfully captures the reentry behavior in the 2D domain.

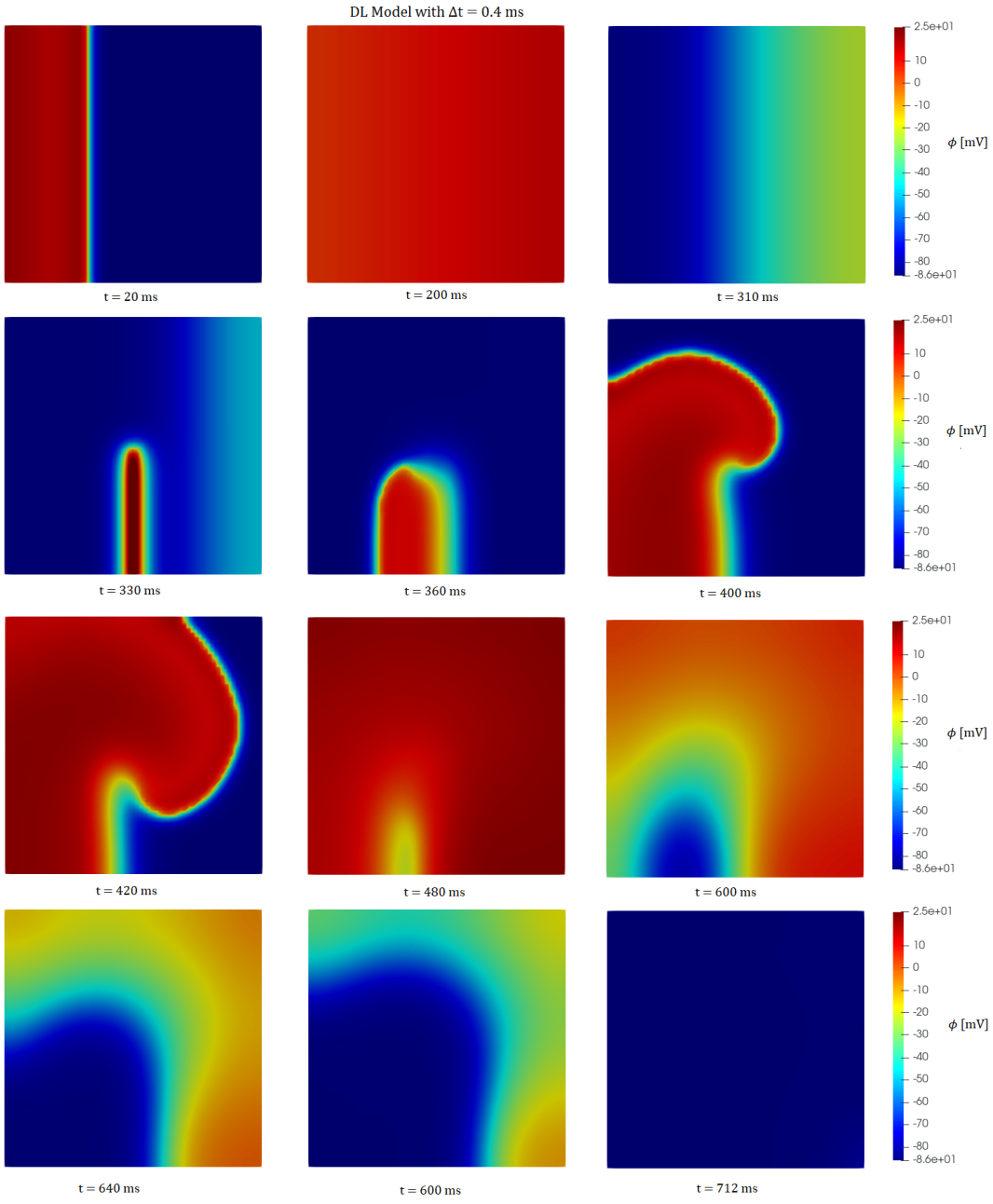

Figure 7.16: Snapshots from the solution obtained with the DL-based algorithm with the time step of $\Delta t = 0.4$ ms with the observed reentry in the domain.

CHAPTER 8

DISCUSSION

The computational modeling of cardiac electrophysiology contributes to both the development of new therapeutic strategies and the improvement of the existing methods. Correct modeling of cardiac electrophysiology is essential for developing treatments and understanding the arrhythmic behavior of the heart and dysfunctions occurring in cardiac tissue. However, even with today's advanced computational tools and mathematical methods, modeling cardiac electrophysiology still requires significant computational time and resources. Therefore, there is still a vital need for robust and efficient methods for solving the electrophysiology problem. Deep learning emerged as a flexible tool that can easily solve scientific problems from various fields; however, its applications in computational science remain relatively limited compared to those in other fields.

This thesis proposes a novel DL-based algorithm for solving cardiac electrophysiology problems. The training of the DL model and its application to different kinds of electrophysiological problems are demonstrated in this study. The architecture of the proposed DL model is designed for fast and accurate predictions for transmembrane potential values and utilized to model electrophysiological behavior for the first time to our knowledge. In this work, the predictive capabilities of DL are utilized to an extent to predict future potential values over large time steps, which are not possible with conventional numerical models. Due to the potential field having high gradient values, specifically in the depolarization stage, the maximum applicable time step size to obtain a solution with numerical methods is very limited. Even with implicit numerical methods, solving the electrophysiology problem demands relatively small time steps to converge successfully. The proposed model is fast to train and deploy,

and the required data can be achieved easily by solving an electrophysiology model in an ODE setting. With the predictor-corrector algorithm, the stimulus term in the ODE problem is associated with the flux term of the PDE, which is another novelty in this work. The proposed approach can be easily extended to different types of PDE problems. Moreover, the proposed DL-based algorithm transforms the original highly nonlinear problem of electrophysiology into a linear problem with a source term provided by the DL module.

The results in Chapter 7 show that the DL-based algorithm can easily model complex problems. Provided solutions by the proposed model are sufficiently accurate and in the physiologically acceptable ranges. The accuracy of the approach faces some challenges due to an increase in the action potential duration and propagation speed of the wavefront. However, since the proposed approach shows the main characteristic behaviors of the action potential, it can be expected to work better with simulations at the organ scale, such as ECG stimulations. Still, the performance of the approach should be improved before cilinical applications since the delay in the wavefront and increase in the action potential duration created large errors in the ventricular domain when error analysis is performed.

The proposed algorithm decreases the computational time in the PDE setting. A major reason for this is the linearization of the PDE problem, which is achieved by the proposed algorithm. As a result of the linearization, a single iteration is required for every time step in the solution stage. The gain in computational time has enormous potential for further improvement. Recent GPUs can process the input much faster than the GPU, which has been used for training and solving the problems in this study. Moreover, newer GPUs have more memory and optimized architectures to allow more predictions to be made simultaneously. Another potential for improvement is the model itself; DL models generally always have room for optimization, which can affect its speed enormously.

Furthermore, it is worth noting that the proposed approach can readily be applied to the DL-based efficient solution reaction-diffusion type parabolic PDEs with highly nonlinear source terms that are inherently nonlinear. For example, the chemo-thermal problem of frontal polymerization can be considered as an immediate example.

REFERENCES

[1] J. Wong, S. Göktepe, and E. Kuhl, "Computational modeling of electrochemical coupling: A novel finite element approach towards ionic models for cardiac electrophysiology," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 45, pp. 3139–3158, 2011.

[2] R. Klabunde, *Cardiovascular physiology concepts*. Lippincott Williams & Wilkins, 2011.

[3] J. D. Kelleher, *Deep learning*. MIT press, 2019.

[4] World Health Organization. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). Last visited on June 2023.

[5] Geneva: World Health Organization, *World Health Statistics 2023 Monitoring health for the SDGs Sustainable Development Goals*. 2023. Licence: CC BY-NC-SA 3.0 IGO.

[6] E. Wilkins, L. Wilson, K. Wickramasinghe, P. Bhatnagar, J. Leal, R. Luengo-Fernandez, R. Burns, M. Rayner, and N. Townsend, *European Cardiovascular Disease Statistics 2017*. Belgium: European Heart Network, February 2017.

[7] V. Raleigh, D. Jefferies, and D. Wellings, *Cardiovascular disease in England: supporting leaders to take actions*. The King's Fund, November 2022.

[8] Centers for Disease Control and Prevention. https://www.cdc.gov/heartdisease/facts.htm. Last visted on June 2023.

[9] C. W. Tsao, A. W. Aday, Z. I. Almarzooq, C. A. Anderson, P. Arora, C. L. Avery, C. M. Baker-Smith, A. Z. Beaton, A. K. Boehme, A. E. Buxton, Y. Commodore-Mensah, M. S. Elkind, K. R. Evenson, C. Eze-Nliam, S. Fugar, G. Generoso, D. G. Heard, S. Hiremath, J. E. Ho, R. Kalani, D. S. Kazi, D. Ko, D. A. Levine, J. Liu, J. Ma, J. W. Magnani, E. D. Michos, M. E. Mussolino, S. D. Navaneethan, N. I. Parikh, R. Poudel, M. Rezk-Hanna, G. A. Roth, N. S. Shah, M. P. St-Onge,

E. L. Thacker, S. S. Virani, J. H. Voeks, N. Y. Wang, N. D. Wong, S. S. Wong, K. Yaffe, and S. S. Martin, "Heart disease and stroke statistics - 2023 update: A report from the american heart association," *Circulation*, vol. 147, pp. E93–E621, January 2023.

[10] N. Sue, W. Laurie, K. Olga, P. Diana, and A. Leib. https://www.heart.org/-/media/Files/Get-Involved/Advocacy/Burden-Report-Technical-Report.pdf. Last visited on June 2023.

[11] OECD, *Cardiovascular Disease and Diabetes: Policies for Better Health and Quality of Care*. 2015.

[12] A. Timmis, P. Vardas, N. Townsend, A. Torbica, H. Katus, D. D. Smedt, C. P. Gale, A. P. Maggioni, S. E. Petersen, R. Huculeci, D. Kazakiewicz, V. de Benito Rubio, B. Ignatiuk, Z. Raisi-Estabragh, A. Pawlak, E. Karagiannidis, R. Treskes, D. Gaita, J. F. Beltrame, A. McConnachie, I. Bardinet, I. Graham, M. Flather, P. Elliott, E. A. Mossialos, F. Weidinger, and S. Achenbach, "European society of cardiology: cardiovascular disease statistics 2021," *European Heart Journal*, vol. 43, pp. 716–799, 2022.

[13] K. H. W. J. ten Tusscher, D. Noble, P. J. Noble, and A. V. Panfilov, "A model for human ventricular tissue," *American Journal of Physiology Heart and Circulatory Physiology*, 2004.

[14] C. H. Luo and Y. Rudy, "A dynamic model of the cardiac ventricular action potential. i. simulations of ionic currents and concentration changes," *Circulation Research*, 1994.

[15] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[16] R. Fitzhugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical Journal*, 1961.

[17] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, "Machine learning in cardiovascular flows modeling: Predicting arterial blood

pressure from non-invasive 4d flow mri data using physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 358, p. 112623, 2020.

[18] J.-M. Kwon, Y. Lee, Y. Lee, S. Lee, and J. Park, "An algorithm based on deep learning for predicting in-hospital cardiac arrest," *Journal of the American Heart Association*, vol. 7, p. e008678, June 2018.

[19] A. Meade and A. Fernandez, "The numerical solution of linear ordinary differential equations by feedforward neural networks," *Mathematical and Computer Modelling*, vol. 19, no. 12, pp. 1–25, 1994.

[20] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[21] M. Deng, T. Meng, J. Cao, S. Wang, J. Zhang, and H. Fan, "Heart sound classification based on improved mfcc features and convolutional recurrent neural networks," *Neural Networks*, vol. 130, pp. 22–32, 2020.

[22] O. Bernard, A. Lalande, C. Zotti, F. Cervenansky, X. Yang, P.-A. Heng, I. Cetin, K. Lekadir, O. Camara, M. A. Gonzalez Ballester, G. Sanroma, S. Napel, S. Petersen, G. Tziritas, E. Grinias, M. Khened, V. A. Kollerathu, G. Krishnamurthi, M.-M. Rohé, X. Pennec, M. Sermesant, F. Isensee, P. Jäger, K. H. Maier-Hein, P. M. Full, I. Wolf, S. Engelhardt, C. F. Baumgartner, L. M. Koch, J. M. Wolterink, I. Išgum, Y. Jang, Y. Hong, J. Patravali, S. Jain, O. Humbert, and P.-M. Jodoin, "Deep learning techniques for automatic mri cardiac multi-structures segmentation and diagnosis: Is the problem solved?," *IEEE Transactions on Medical Imaging*, vol. 37, no. 11, pp. 2514–2525, 2018.

[23] J. Schlemper, J. Caballero, J. Hajnal, A. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for dynamic mr image reconstruction," *IEEE Transactions on Medical Imaging*, vol. PP, 04 2017.

[24] G. A. Bello, T. J. W. Dawes, J. Duan, C. Biffi, A. de Marvao, L. S. G. E. Howard, J. S. R. Gibbs, M. R. Wilkins, S. A. Cook, D. Rueckert, and D. P. O'Regan,

"Deep-learning cardiac motion analysis for human survival prediction," *Nature Machine Intelligence*, vol. 1, pp. 95–104, February 2019.

[25] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, pp. 65–69, Jan 2019.

[26] A. H. Ribeiro, M. H. Ribeiro, G. M. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. S. Ferreira, C. R. Andersson, P. W. Macfarlane, W. M. Jr., T. B. Schön, and A. L. P. Ribeiro, "Automatic diagnosis of the 12-lead ecg using a deep neural network," *Nature Communications*, vol. 11, p. 1760, April 2020.

[27] A. Isin and S. Ozdalili, "Cardiac arrhythmia detection using deep learning," *Procedia Computer Science*, vol. 120, pp. 268–275, 2017. 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary.

[28] G. M., V. Ravi, S. V, G. E.A, and S. K.P, "Explainable deep learning-based approach for multilabel classification of electrocardiogram," *IEEE Transactions on Engineering Management*, vol. 70, no. 8, pp. 2787–2799, 2023.

[29] U. Baloglu, M. Talo, O. Yildirim, R. S. Tan, and U. R. Acharya, "Classification of myocardial infarction with multi-lead ecg signals and deep cnn," *Pattern Recognition Letters*, vol. 122, 02 2019.

[30] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," *Journal of Machine Learning Research*, vol. 19, no. 25, pp. 1–24, 2018.

[31] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[32] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, "Stiff neural ordinary differential equations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, 09 2021.

[33] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," *Journal of Computational Physics*, vol. 395, pp. 620–635, 06 2019.

[34] W.-H. Su, C.-S. Chou, and D. Xiu, "Deep learning of biological models from data: Applications to ode models," *Bulletin of mathematical biology*, vol. 83, p. 19, January 2021.

[35] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, pp. 987–1000, 09 1998.

[36] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. 4, p. e1602614, 2017.

[37] W. E and B. Yu, "The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems," *Communications in Mathematics and Statistics*, vol. 6, pp. 1–12, March 2018.

[38] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.

[39] Y. Guo, X. Cao, B. Liu, and M. Gao, "Solving partial differential equations using deep learning and physical constraints," *Applied Sciences*, vol. 10, no. 17, 2020.

[40] M. A. Finzi, A. Potapczynski, M. Choptuik, and A. G. Wilson, "A stable and scalable method for solving initial value PDEs with neural networks," in *The Eleventh International Conference on Learning Representations*, 2023.

[41] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 34, pp. 8505–8510, 2018.

[42] A. Li, R. Chen, A. B. Farimani, and Y. J. Zhang, "Reaction diffusion system prediction based on convolutional neural network," *Scientific Reports*, vol. 10, p. 3894, Mar 2020.

[43] C. Rao, P. Ren, Q. Wang, O. Buyukozturk, H. Sun, and Y. Liu, "Encoding physics to learn reaction–diffusion processes," *Nature Machine Intelligence*, vol. 5, pp. 765–779, 7 2023.

[44] B. Zakeri, M. Khashehchi, S. Samsam, A. Tayebi, and A. Rezaei, "Solving partial differential equations by a supervised learning technique, applied for the reaction–diffusion equation," *SN Applied Sciences*, vol. 1, p. 1589, November 2019.

[45] G. Peng, M. Alber, A. Buganza Tepole, W. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. Lytton, P. Perdikaris, L. Petzold, and E. Kuhl, "Multiscale modeling meets machine learning: What can we learn?," *Archives of Computational Methods in Engineering*, vol. 28, pp. 1–21, 02 2020.

[46] Y. Wang, S. W. Cheung, E. T. Chung, Y. Efendiev, and M. Wang, "Deep multiscale model learning," *Journal of Computational Physics*, vol. 406, p. 109071, 2020.

[47] J. Xie and B. Yao, "Physics-constrained deep learning for robust inverse ecg modeling," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 151–166, 2023.

[48] L. Li, J. Camps, A. Banerjee, M. Beetz, B. Rodriguez, and V. Grau, "Deep computational model for the inference of ventricular activation properties," in *Statistical Atlases and Computational Models of the Heart. Regular and CMRx-Motion Challenge Papers* (O. Camara, E. Puyol-Antón, C. Qin, M. Sermesant, A. Suinesiaputra, S. Wang, and A. Young, eds.), pp. 369–380, Springer Nature Switzerland, 2022.

[49] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, "Physics-informed neural networks for cardiac activation mapping," *Frontiers in Physics*, vol. 8, 2020.

[50] C. D. Cantwell, Y. Mohamied, K. N. Tzortzis, S. Garasto, C. Houston, R. A. Chowdhury, F. S. Ng, A. A. Bharath, and N. S. Peters, "Rethinking multiscale cardiac electrophysiology with machine learning and predictive modelling," *Computers in Biology and Medicine*, vol. 104, pp. 339–351, 2019.

[51] J. Xie and B. Yao, "Physics-constrained deep active learning for spatiotemporal modeling of cardiac electrodynamics," *Computers in Biology and Medicine*, vol. 146, p. 105586, 2022.

[52] C. Herrero Martin, A. Oved, R. A. Chowdhury, E. Ullmann, N. S. Peters, A. A. Bharath, and M. Varela, "EP-PINNs: Cardiac electrophysiology characterisation using physics-informed neural networks," *Frontiers in Cardiovascular Medicine*, vol. 8, 2022.

[53] I. Nazarov, I. Olakorede, A. Qureshi, S. Ogbomo-Harmitt, and O. Aslanidi, "Physics-informed fully connected and recurrent neural networks for cardiac electrophysiology modelling," in *2022 Computing in Cardiology (CinC)*, vol. 498, pp. 1–4, 2022.

[54] S. Court and K. Kunisch, "Design of the monodomain model by artificial neural networks," *Discrete and Continuous Dynamical Systems*, vol. 42, no. 12, pp. 6031–6061, 2022.

[55] S. Herzog, F. Wörgötter, and U. Parlitz, "Data-driven modeling and prediction of complex spatio-temporal dynamics in excitable media," *Frontiers in Applied Mathematics and Statistics*, vol. 4, 12 2018.

[56] I. Ayed, N. Cedilnik, P. Gallinari, and M. Sermesant, "EP-Net: Learning cardiac electrophysiology models for physiology-based constraints in data-driven predictions," in *Functional Imaging and Modeling of the Heart* (Y. Coudière, V. Ozenne, E. Vigmond, and N. Zemzemi, eds.), (Cham), pp. 55–63, Springer International Publishing, 2019.

[57] V. Kashtanova, I. Ayed, N. Cedilnik, P. Gallinari, and M. Sermesant, "EP-Net 2.0: Out-of-domain generalisation for deep learning models of cardiac electrophysiology," in *FIMH 2021 - 11th International Conference on Functional Imaging and Modeling of the Heart*, vol. 12738 of *Lecture Notes in Computer Science*, (Stanford, CA (virtual), United States), pp. 482–492, Springer International Publishing, June 2021.

[58] V. Kashtanova, I. Ayed, A. Arrieula, M. Potse, P. Gallinari, and M. Sermesant, "Deep learning for model correction in cardiac electrophysiological imaging,"

in *MIDL 2022 - Medical Imaging with Deep Learning*, (Zurich, Switzerland), 7 2022.

[59] V. Kashtanova, M. Pop, I. Ayed, P. Gallinari, and M. Sermesant, "APHYN-EP: Physics-based deep learning framework to learn and forecast cardiac electrophysiology dynamics," in *STACOM 2022 - 13th Workhop on Statistical Atlases and Computational Modelling of the Heart*, (Singapore, Singapore), Sept. 2022.

[60] S. Fresca, L. Dede', and A. Manzoni, "A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes," *Journal of Scientific Computing*, vol. 87, p. 61, April 12 2021.

[61] S. Fresca, A. Manzoni, L. Dedè, and A. Quarteroni, "Deep learning-based reduced order models in cardiac electrophysiology," *PLoS One*, vol. 15, no. 10, 2020.

[62] S. Fresca, A. Manzoni, L. Dedè, and A. Quarteroni, "Pod-enhanced deep learning-based reduced order models for the real-time simulation of cardiac electrophysiology in the left atrium," *Frontiers in Physiology*, vol. 12, 2021.

[63] B. Gonçalves and M. Deo, "Modeling cardiac cell biophysics using long-short-term memory networks," in *2022 Annual Modeling and Simulation Conference (ANNSIM)*, pp. 341–350, 2022.

[64] S. Shahi, C. D. Marcotte, C. J. Herndon, F. H. Fenton, Y. Shiferaw, and E. M. Cherry, "Long-time prediction of arrhythmic cardiac action potentials using recurrent neural networks and reservoir computing," *Frontiers in Physiology*, vol. 12, p. 734178, Sep 2021.

[65] S. Shahi, F. H. Fenton, and E. M. Cherry, "A machine-learning approach for long-term prediction of experimental cardiac action potential time series using an autoencoder and echo state networks," *Chaos*, vol. 32, p. 063117, Jun 2022.

[66] M. Courtemanche, R. J. Ramirez, and S. Nattel, "Ionic mechanisms underlying human atrial action potential properties: Insights from a mathematical model," *The American Journal of Physiology*, 1998.

[67] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *Proceedings of the IRE*, vol. 50, no. 10, pp. 2061–2070, 1962.

[68] R. R. Aliev and A. V. Panfilov, "A simple two-variable model of cardiac excitation," *Chaos, Solitons & Fractals*, 1996.

[69] C. C. Mitchell and D. G. Schaeffer, "A two-current model for the dynamics of cardiac membrane," *Bulletin of Mathematical Biology*, 2003.

[70] K. H. W. J. ten Tusscher and A. V. Panfilov, "Alternans and spiral breakup in a human ventricular tissue model," *American Journal of Physiology Heart and Circulatory Physiology*, 2006.

[71] S. Krishnamoorthi, M. Sarkar, and W. S. Klug, "Numerical quadrature and operator splitting in finite element methods for cardiac electrophysiology," *International journal for numerical methods in biomedical engineering*, pp. 1243–1266, 2013.

[72] D. J N Reddy, *An Introduction to the Finite Element Method*. McGraw-Hill Education, 2005.

[73] F. Chollet, *Deep learning with Python*. Simon and Schuster, 2021.

[74] T. Mansi, T. Passerini, and D. Comaniciu, *Artificial intelligence for computational modeling of the heart*. Academic Press, 2019.

[75] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[76] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[77] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014.

[78] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[79] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[80] D. Kingma and L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[81] R. L. Taylor, "Feap-a finite element analysis program," 2014.

[82] M. Kotikanyadanam, S. Göktepe, and E. Kuhl, "Computational modeling of electrocardiograms: A finite element approach toward cardiac excitation," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 26, no. 5, pp. 524–533, 2010.